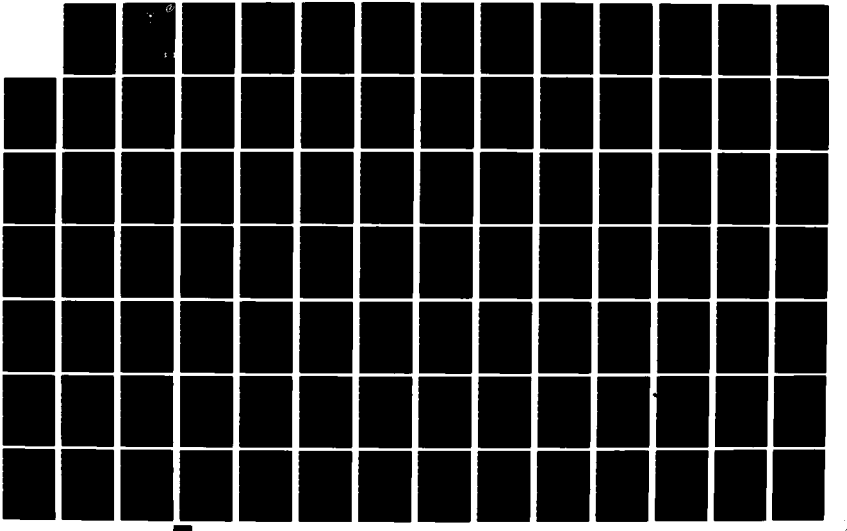
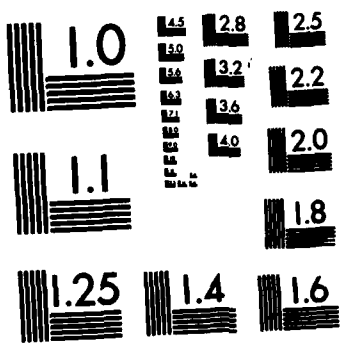
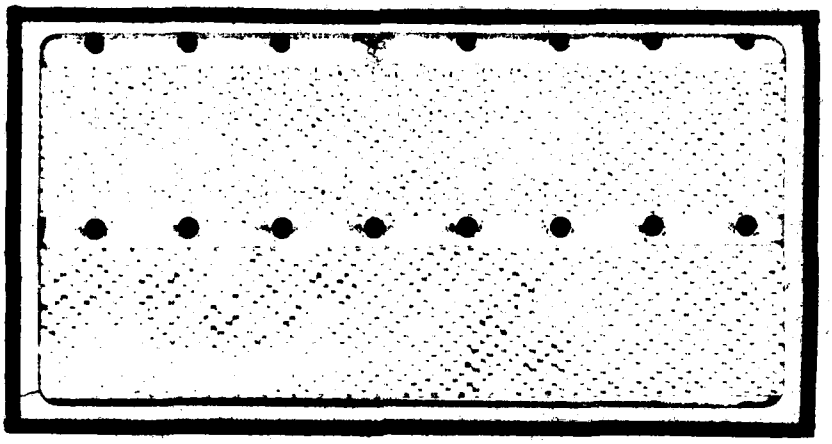
AD-A124 908 A BACKEND DATA BASE SYSTEM FOR THE EGLIN COMPUTER NETWORK(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING J L SELF NOV 82 1/2

UNCLASSIFIED AFIT/GCS/EE/82D-32 F/G 9/2 NL







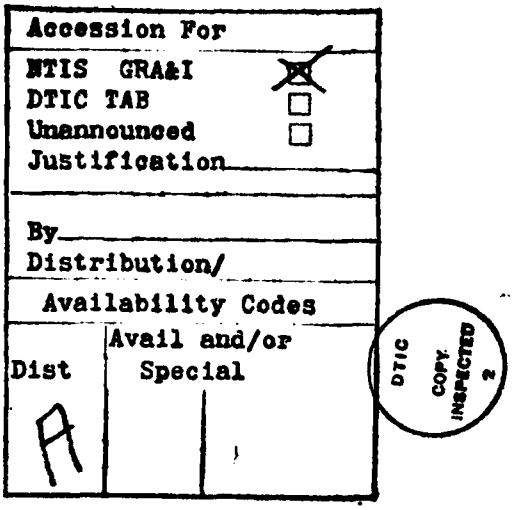
DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY (ATC)

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCS/EE/82D-32



A BACKEND DATA BASE SYSTEM FOR THE EGLIN COMPUTER NETWORK

THESIS

AFIT/GCS/EE/82D-32 Joseph L. Self
Capt USAF

Approved for public release; distribution unlimited

AFIT/GCS/EE/82D-32

A BACKEND DATABASE SYSTEM FOR THE EGLIN COMPUTER NETWORK

THESIS

Presented To The Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

in Partial Fulfillment for the Degree of

Master of Science

by

Joseph L. Self, B.S., M.B.A

Capt USAF

Graduate Computer Science

November 1982

Preface

Backend database management systems have shown great potential for improvement over conventional database management systems. The Eglin Computer Network (ECONET) expects an increased workload in its database applications and was therefore interested in examining a backend database system. This thesis attempts to evaluate the ECONET and its future database needs and to present a backend database design as a possible approach to meet those needs.

I would like to thank my thesis advisor, Major Michael Varrieur, for his guidance and assistance in this effort. Also my other committee members, Dr. Thomas Hartrum and Dr. Henry Potoczny provided many constructive comments. Professor Daniel Reynolds also deserves special mention for his help in debugging numerous simulation programs. Finally, I thank my wife, Linda, for her support and patience during these past 18 months. She made it all worthwhile.

CONTENTS

LIST OF FIGURES

LIST OF FIGURES (Continued)

Abstract

The Directorate of Computer Sciences at Eglin AFB is in the process of upgrading its computer system to meet increased future requirements. Their upgrade includes a revision of the data base query capabilities to handle a projected larger on-line data base requirement. A backend data base computer system was investigated as a possibility for meeting the anticipated increase in data base queries.

A systems analysis of the Eglin Computer Network's (ECONET) current data base query workload was accomplished to determine a baseline for future workload projections. A modified Multi-backend Data Base System (MDBS - developed by David Hsiao) design was then incorporated into the ECONET and simulated with numerous structural and parametric variations. The parametric variations were to measure performance changes due to changes in workloads and the types of jobs processed. The structural variations were made to measure performance changes caused by changes in hardware configurations.

The simulation results were used to select a final backend configuration that, hopefully, would best meet the requirements of data base processing on the ECONET. A MDBS-like configuration with three backend processors was chosen.

I. INTRODUCTION

Background

During recent years the use of Database Management Systems (DBMS) by commercial and government organizations has increased greatly. People using computers have come to expect fast turnaround and easy access to large volumes of data. In many cases these people were not disappointed. Database management systems, for the most part, have provided quick turnaround and allowed workers to increase their productivity substantially. More recently however, many organizations have experienced degradations in their DBMS's performance due to a higher than expected use of the system. As more people discover the power and simplicity of the multitude of DBMS's offered on the market today, the more saturated those systems will become.

To solve this saturation problem, several approaches have been taken. Probably the most widely used approach until recently has focused on the DBMS software itself. DBMS algorithms and data model structures are continually under review. Designer's are always looking for methods to increase efficiency and decrease processing time. A number of researchers are studying the area of query optimization and have shown that with a little thought, response time for a query can be substantially reduced with minor (or sometimes not so minor) rearrangement. Relational data models, in particular, have recently spawned volumes of research in the area query optimization.

Another approach to the saturation problem has been to simply upgrade the computer mainframe upon which the DBMS resides. By allowing the DBMS access to faster processors and I/O subsystems the response time for DBMS transactions can be improved. This solution, however, can be very expensive. The purchase of a large computer can put a sizable dent in even a large organization's budget. Also this solution often only postpones the problem.

Still another approach to the saturation problem that has gotten considerable attention during the past few years has been the backend DBMS. Basically this approach suggests offloading the DBMS from the mainframe computer on to a directly attached minicomputer. This approach allows several parallel operations and can be considerably cheaper than upgrading to a larger mainframe computer. The concept also lends itself easily to numerous related designs such as multiple backend processors per host computer, multiple hosts per backend processor, or many other configurations. Because of its numerous potentials and advantages, the backend DBMS concept is the approach taken in this thesis.

Statement of the Problem

The purpose of this thesis is to solve three related problems. First is to analyze the computer system at Eglin AFB with a view towards incorporating a backend DBMS. The Eglin Computer Network (ECONET) expects future reduced response times for DBMS queries during peak hours because of saturation. Second is to include a backend DBMS design into

the ECONET and analyze its performance using simulation techniques. Third is to make recommendations on the design of a backend DBMS in the ECONET using the results of the simulation.

Scope

The scope of this thesis is to solve the three problems defined in the "statement of the problem" section and to present those solutions in a document that is beneficial to Eglin personnel.

Approach

Whenever one is attempting a large project such as a thesis, a well defined approach helps ensure successful completion. That plan is described in this section.

First an extensive literature search was done to gain a better understanding of the topic and to provide a starting point for continued research throughout the thesis effort. The literature search included investigation into topics surrounding the backend DBMS concepts and the Eglin Computer Network. This portion of the thesis effort was very important because it not only provided material for background information, but also provided a foundation for the eventual design of a backend DBMS for the ECONET.

After conducting the initial review of the literature, an analysis of the ECONET was made to provide input for development of a simulation model of the ECONET with a backend DBMS. Several methods of gathering information were used including: interviews, review of ECONET management

tools, and a study of available system statistics and workload studies. The focus was upon how the quantity and complexity of queries effected the responsiveness of the current DBMS and in turn how the productivity of its users was affected.

The next step in the thesis was the development of a backend DBMS that could be incorporated into the ECONET. The information gained from the systems analysis was used to determine the backend configuration itself and also to develop the baseline parametric models necessary to simulate the ECONET with a backend DBMS.

Finally the simulation was used to determine an optimum configuration for the backend DBMS portion of the ECONET. Additional recommendations will also be made depending upon the simulation results.

Overview of the Thesis

The structure of this thesis follows the approach that was taken in the investigation. Chapter II presents a background review of backend database computer systems and of the Eglin Computer Network. Chapter III describes the systems analysis done at Eglin and how parametric models were developed from this analysis for simulation of a backend computer system. Chapter IV is a description of the backend database computer system that was simulated and how it fits into the ECONET. Chapter V is a presentation of the simulation and its results. Finally, Chapter VI summarizes the thesis investigation and presents recommendations.

II. BACKGROUND

Introduction

To ensure that the reader understands the logic behind several architectural decisions made in the design of a backend database computer for the Eglin Computer Network, a background chapter has been included in this thesis. Two major sections are presented in this background chapter. Because of the importance of the many factors that contribute to the design of a backend database computer a good bit of detail will be included in each of the two sections.

In the first section, the Eglin Computer Network will be discussed. The current configuration and its problems, (also mentioned in the Systems Analysis Section) along with the planned modular upgrade approach will be presented. Also a more detailed description of the final "desired configuration" will be presented with an explanation of each major module in the system.

In the second section, backend database computer concepts are discussed. Here the basic concepts of the backend database computer will be presented along with its advantages and disadvantages. Once the basic concepts have been presented, a finer distinction between the hardware implementation of the backend database computer (called Database Computer, DBC) and the software database computer will be made. A hardware implementation of the backend concept uses specialized processors developed specifically

for database processing. There is little or no software included in its design. A software implementation uses general purposes processors (usually minicomputers) along with specialized software to assist in database processing. In addition to discussing the general principals of these two implementations, notable prototypes of each will be discussed in depth. This section is extremely important because it provides the basis for several design decisions.

The Eglin Computer Network (ECONET) (Directorate of Computer Sciences Planning Information, 1981)

The Directorate of Computer Sciences (KR) provides centralized management of computational resources for the Armament Division (AD). KR advises the AD commander, staff, and other agencies on current and projected capabilities. It provides mathematical, computational, and data reduction services in support of both scientific and business applications. The business applications are the primary focus of this thesis, but other applications must also be considered, because in many cases they share resources with the business applications.

Current Configuration

The current configuration of the ECONET is shown in Figure 1. The current system is really two separate systems, one processing classified data and the other processing unclassified data. AD/KR currently uses two CDC 6600 computers, one CDC Cyber 176 computer, one IBM 360/65

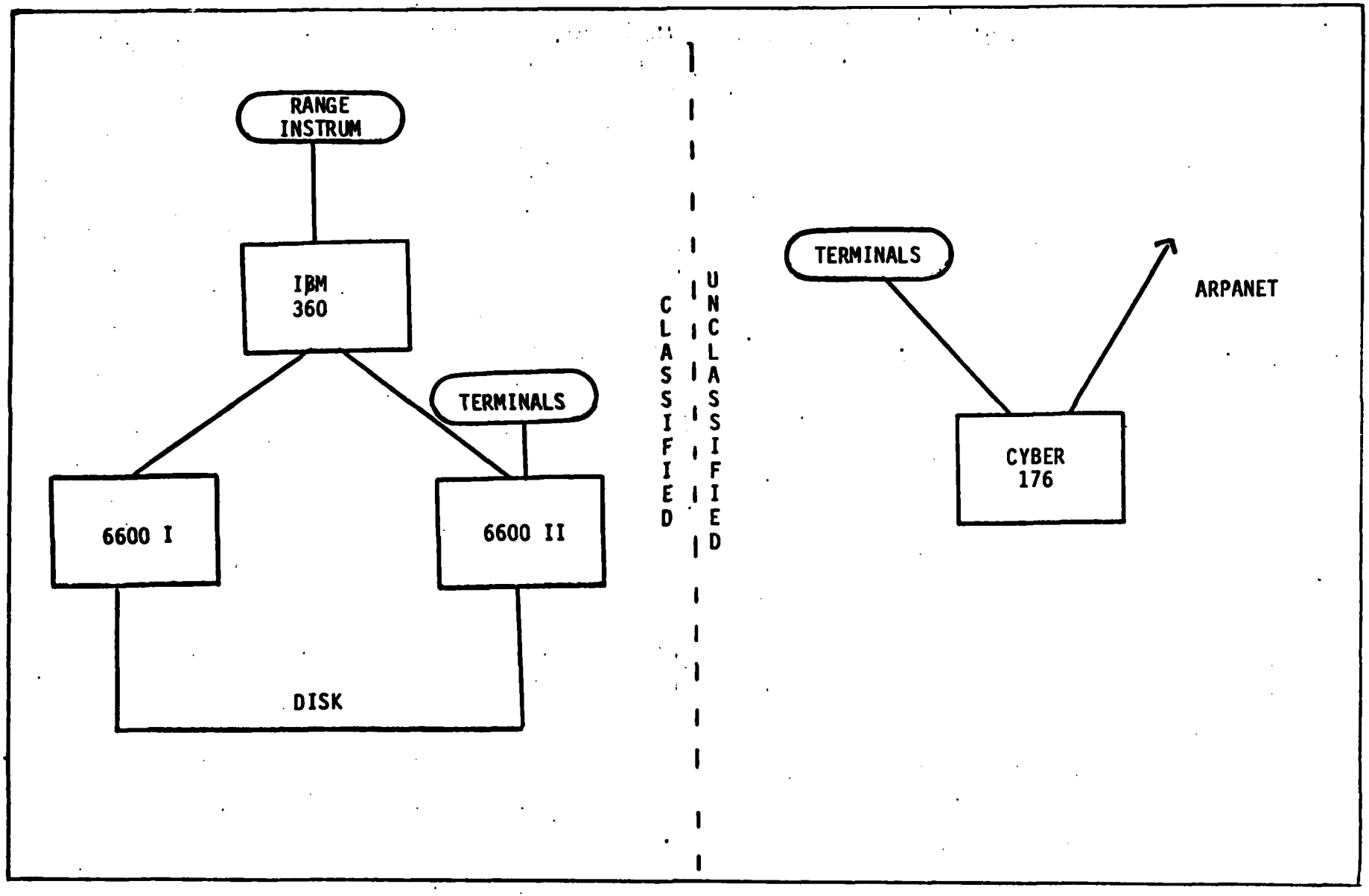


Figure 1. Current Configuration
(Directorate of Computer Sciences Planning Information, 1981: 19)

computer and approximately 50 smaller scale mini and micro computers to support both the scientific and business applications previously mentioned.

The two CDC 6600 computers are connected via a shared disk subsystem to appear to the user as a single multi-mainframe system. One of the systems supports classified interactive terminals and classified batch processing while the other system shares the classified batch processing along with supporting real time activity. The CDC Cyber 176 supports unclassified batch and interactive processing. The current database management system is System 2000 (supported by Intel Corp) and resides on the CDC Cyber 176.

The IBM 360/65 is connected to the CDC 6600's via a locally built channel-to-channel interface, and to several PDP-11's through standard Digital Equipment Corporation (DEC) interfaces. The PDP-11's (range instrumentation) are connected either directly or indirectly to several terminals and microprocessors located throughout the Eglin test ranges. The PDP-11's and microprocessors serve to collect and compare incoming data streams and to format and drive the various output display terminals. The IBM 360/65 supports some small (CPU) real time applications and serves as a front-end to the CDC 6600 to support most large scale real time applications.

Several apparent and not so apparent problems are associated with the system's current configuration. First and most noticable, is the lack of a communications path

between the classified and unclassified systems. Classified customers are restricted from obtaining access to unclassified files on the CDC Cyber 176. Second, the current configuration lacks flexibility and redundancy in mainframe-to-mainframe communication paths. Third, current trends in user demands for service indicate that much higher levels of interactive terminal support will be required throughout the 1980's (Directorate of Computer Sciences Planning Information, 1981: 10). Work load studies conducted by KR personnel indicate that by FY85 the system workload will have doubled. This increase in workload poses a potential threat as the current system is near saturation. Fourth, database management functions on the CDC Cyber 176 are tying up an inordinate amount of processor and I/O time. The CDC Cyber 176 was primarily built as a "number cruncher", as a result of this design its interactive and database handling capabilities are marginal. Fifth, and finally, the users of the system would be severely hampered by an extended conversion period to another configuration. In order to solve this problem AD/KR has started a "modular upgrade approach".

Modular Upgrade Approach

With this method of system upgrade individual functional modules are independently and competitively upgraded with a minimum of impact on the computer center customers. Only functionally deficient modules will be upgraded.

Requirements for Upgrade

AD/KR has several requirements and goals for the planned upgrade of their computer system. The primary goal is to provide a flexible computer complex capable of meeting the increasing demands for state-of-the-art computer service to support all facets of the AD mission. To meet this goal they require that the new configuration has the following characteristics:

Interactive - computer terminals must be interactive and easily accessible to users

Responsive - response times must be short to maximize productivity

On-Line - operator intervention must not be required

Distributed - functions should be performed at the lowest appropriate level. Services not available locally should be accessible through nationwide networking.

Secure - classified and unclassified processing should be available on one system.

Modular - in order to support a non-uniform growth pattern, functional units must be upgraded or removed with minimal system interference.

Reliable - redundancy and recoverability must be provided at all system levels.

Future Configurations

As previously mentioned, the modular upgrade approach allows a non-traumatic piece-by-piece replacement of functionally deficient modules and the addition of new modules that may be needed to meet the requirement just stated. A general overview of the planned system as it progresses through the stages of upgrade will now be discussed. This is important because it leads us to the system configuration that this thesis is concerned with, one that includes a backend database computer. The implementation of the final configuration shown (Figure 4) depends upon the availability of a certified multi-level secure operating system and hopefully an efficient database computer. The multi-level secure operating system will allow combined processing of classified and unclassified data while the backend database computer will provide several other advantages to be mentioned later in this chapter.

October 1982 Configuration

Figure 2 shows the first step beyond the current configuration. On the unclassified system, the most notable additions are the inter-computer bus and the interactive front-ends. The inter-computer bus, when

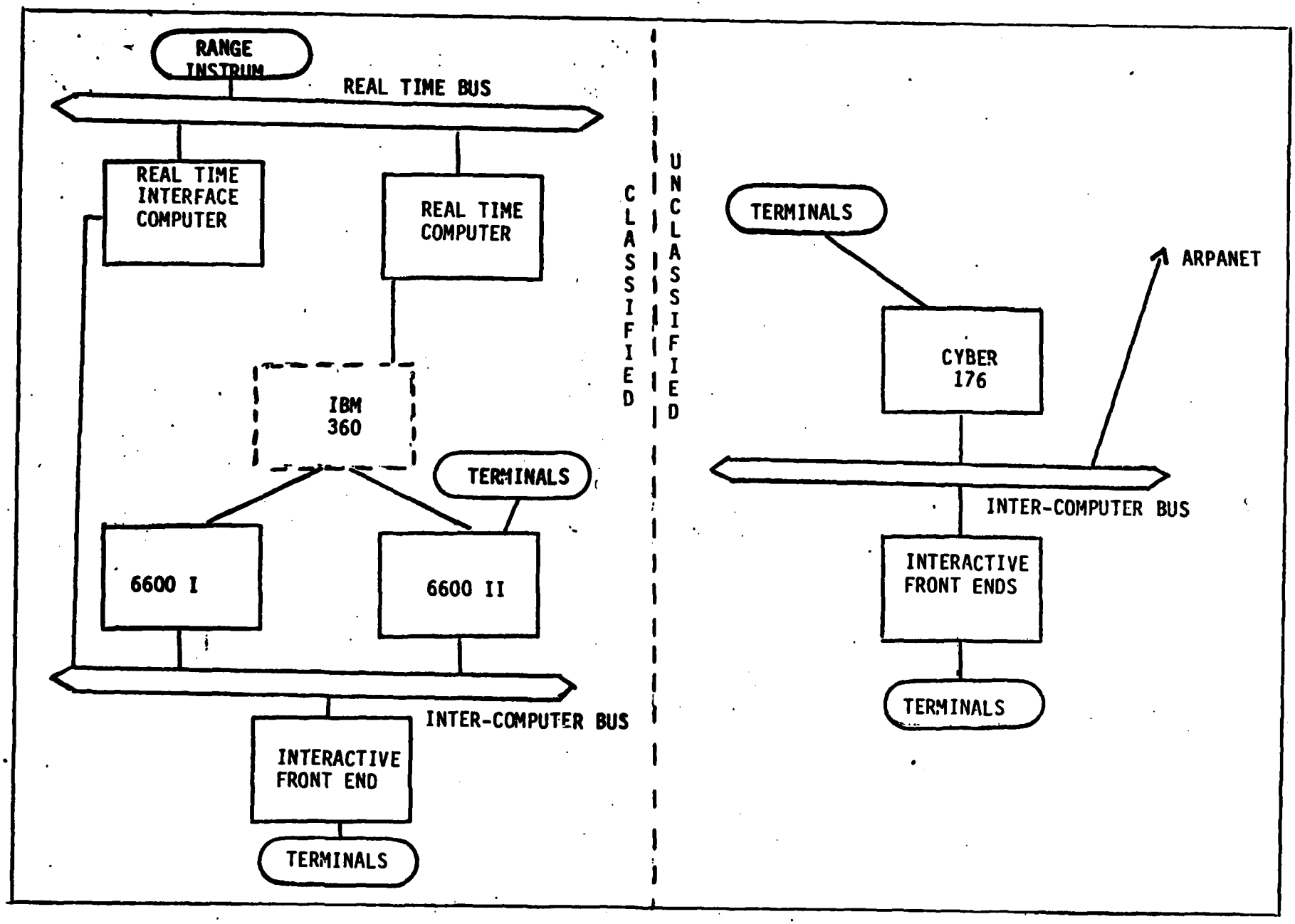


Figure 2 October 82 Configuration
(Directorate of Computer Sciences Planning Information, 1981: 20)

completely implemented (June 1982), will provide very high speed computer-to-computer connections and add much needed flexibility and growth potential. The interactive front-ends (IFE) will be VAX 11/780's. This addition will provide interconnected computer systems to meet the growth in demand for time sharing until FY87 and will remove a majority of the processing load from the CDC Cyber 176 and CDC 6600's and allow the CDC Cyber 176 to perform the function it was designed for, namely number crunching.

On the classified sytem a real time computer is to replace the IBM 360/65, and IFE's are also added here to handle the increased terminal load. Note, however, that the bus does not cross between the classified and unclassified system and there is also no change in the database management configuration.

June 1983 Configuration

By June 1983 the system configuration shown in Figure 3 should be in place. As can be seen more IFE's have been added to handle the increased terminal load on the unclassified system and also a mass storage module has been added. As currently planned, this mass storage system will contain at least two independent processors, each having independent connections to the inter-computer bus. The mass storage system is to be the file repository for all files in the current tape library and permanent file subsystems. More detail will be given on the mass storage system and the other new modules later in this chapter.

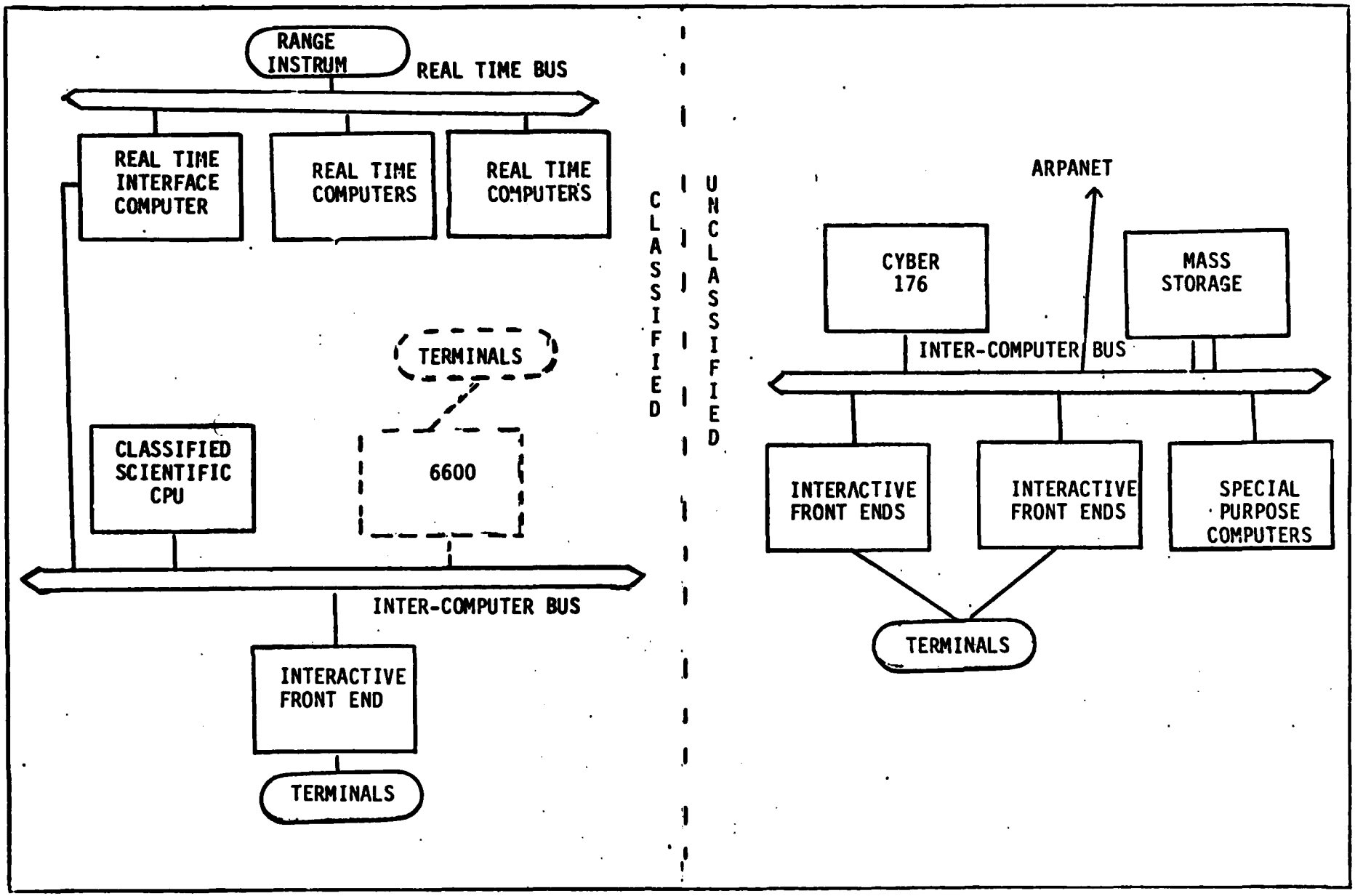


Figure 3 June 83 Configuration
(Directorate of Computer Sciences Planning Information, 1981: 22)

The "special purposes computers" block on the unclassified system is shown to allow for future addition of special purpose functional modules. These modules may be added to improve specific application performance or to remove and isolate particular functions from the rest of the network. These special purpose computers may be interfaced directly to the inter-computer bus or indirectly through interactive front-ends.

On the June 1983 classified system (note these still are two separate systems; classified and unclassified) a new scientific mainframe computer will have replaced the two CDC 6600's. This will be done because the number, duration and complexity of real time missions, batch computer simulations, data reduction, and scientific information retrieval programs will have increased to the point where a more powerful central processor will be required. (Directorate of Computer Sciences Planning Information, 1981: 41). Computer workload projections (done by Eglin personnel) indicate a real time instantaneous peak demand and a batch saturation that can only be relieved by an enhanced mainframe. Also note that the new scientific computer and the real time interface computer have eliminated the need for the IBM 360/65 real time front-end.

Final Configuration

The final configuration of the modular upgrade approach is shown in Figure 4. In this configuration the

multi-level secure operating system is in place, allowing mixed classified and unclassified processing. Access to classified data is controlled by limiting access to the separate classified IFE's and by the operating sytem. As will be discussed later, the system's security can also be enhanced by the backend database computer.

For the remainder of this section our discussion of the modules in Figure 4 will be limited to those modules representing the interactive front-ends, the high speed bus, the mass storage system, and the backend database computer (not shown on Figure 4).

Interactive Front-ends (IFE) (DAR 80-10)

As Figure 5 indicates, the number of timesharing users of the ECONET has steadily increased and will continue to increase into the future. The current configuration simply can not handle this increase in terminal usage. The solution proposed by Eglin personnel is a local interactive network of medium scale computers, each handling approximately 50 terminals. The computers are to be interconnected at channel speeds so that a user at any one of the front-end system can access the resources of any other front-end. The classified and unclassified IFE's however, will be separated. The entire front-end network will be interconnected to the central mass storage system and large processor via a high speed channel. Each system will be configured so that in an emergency it would be capable of processing in a stand alone fashion. All of the

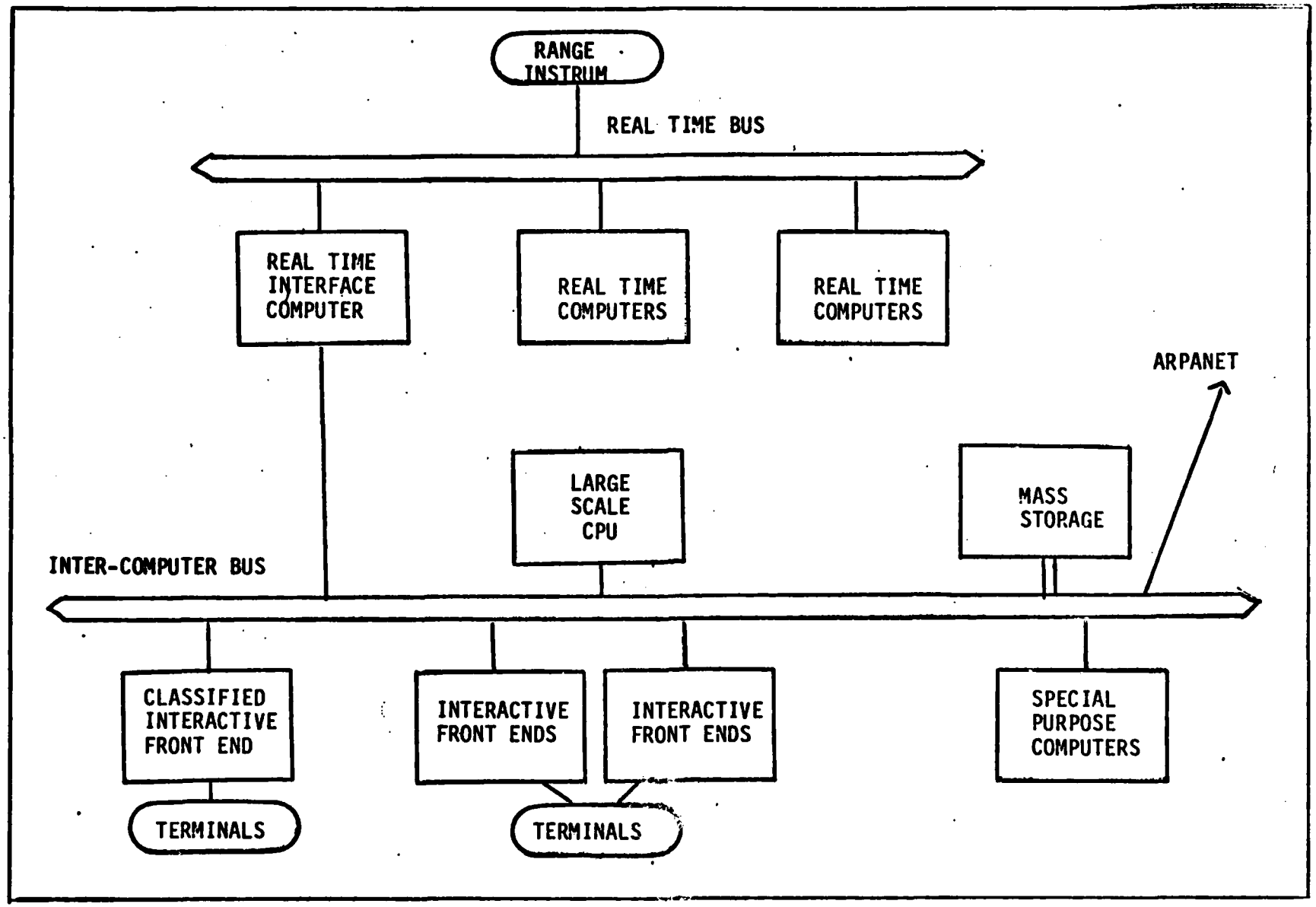


Figure 4 June 86 Configuration
(Directorate of Computer Sciences Planning Information, 1981: 23)

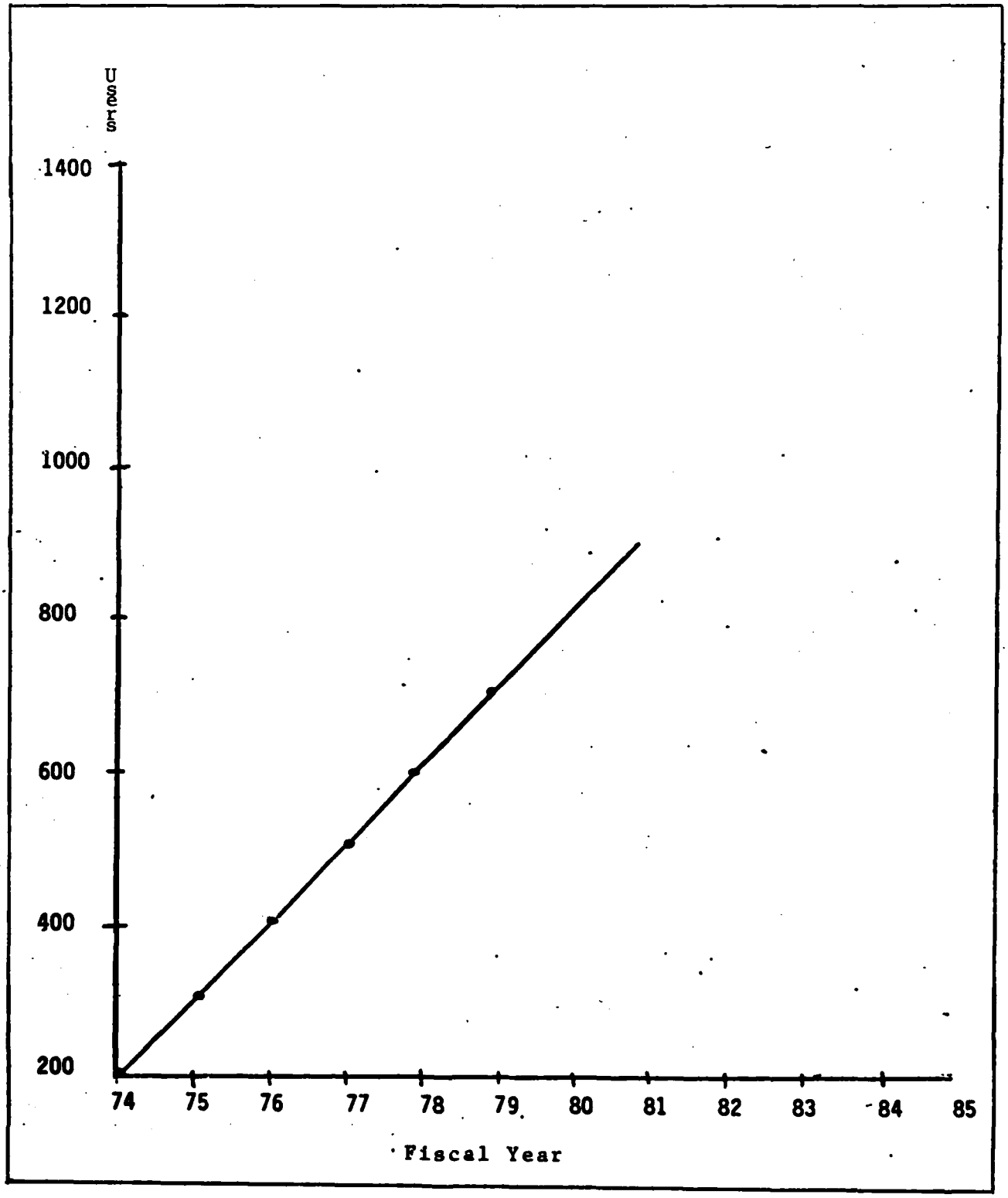


Figure 5 Time Sharing Users
(DAR 80-10, 1979)

IFE's will have a timesharing oriented operating system and have all the tools necessary to support a timesharing user to include:

1) Editors
2) Graphics Processors
3) Remote batch processors
4) Text processors
5) Mail system
6) On-line documentation
7) Other normal program development tools

It is envisioned that many tasks will be handled completely by the front-end computers. Concerning database applications, for example, a good deal of query formatting and optimization could be done on the IFE before being sent to the backend database computer. There will of course be some applications that require the power of the larger central processor. This also will be possible by connecting the front-ends to the larger system via the high speed bus.

High Speed Bus

Figure 6, gives a somewhat detailed pictorial representation of the ECONET high speed bus and the connection of the IFE's to it. The inter-computer bus will provide a reliable, commonly accessible high speed channel to interconnect various general and special purpose

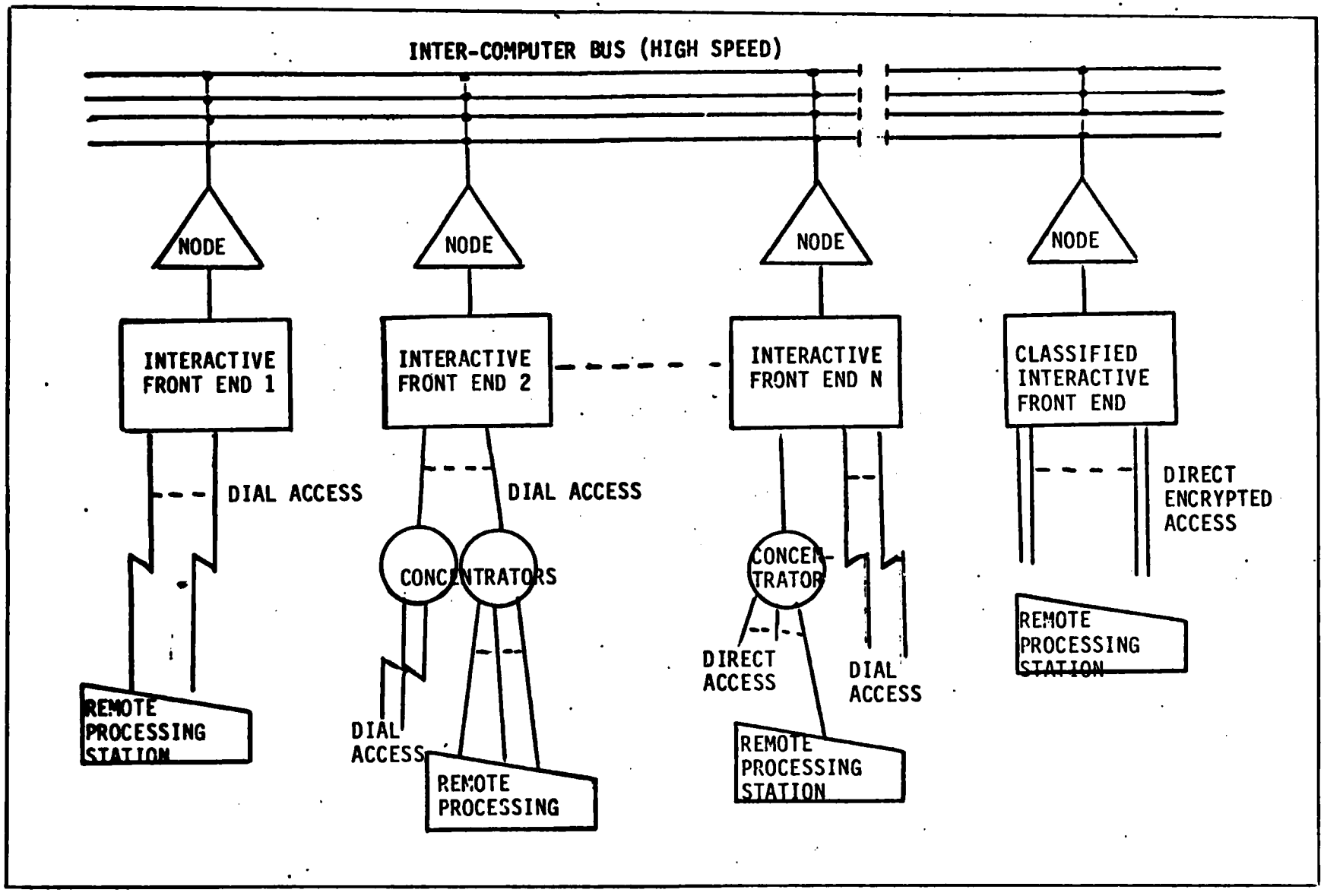


Figure 6 . Interactive Front End Computers (DAR ADTC-79-29: 21)

computer systems manufactured by several different vendors. The acquisition of this communication device should help balance the loads between different machines, reduce data file duplication and reduce processor waste due to lower level tasks being performed by expensive processors. The bus is to contain all necessary interface hardware, firmware and software and will exhibit the following characteristics:

1) Task communication path will support at least 40 megabits/second transmission rate up to 1000 ft
2) Task interface node will be capable of accessing at least 4 independent paths
3) Task interface node will be user-programmable
4) The failure of any one node will not interrupt the operation of any other node on the channel
5) The interface nodes will contain transmission parity error detecting and correction mechanisms
6) The channel will initially support seven nodes, but will be expandable to at least 64 nodes (see Figure 7)

Mass Storage System (DAR 80-11, 1980)

The proposed mass storage system is shown in Figure 8. Tape users on the ECONET have historically been plagued with poor tape performance and a high rate of wasted computer time and manpower due to tape parity errors. Also with the already mentioned increase in terminal usage, a larger amount of on-line storage is needed. Eglin

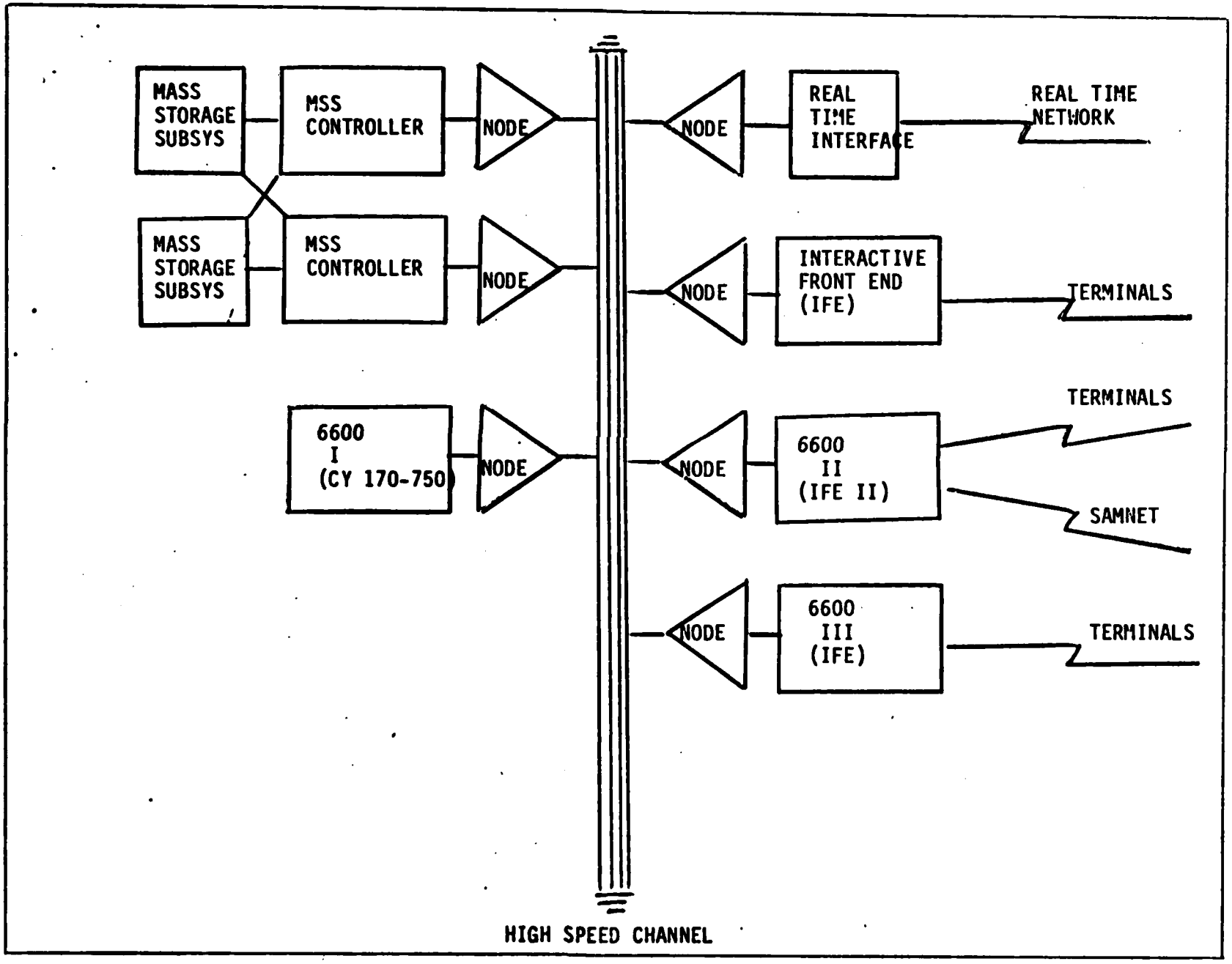


Figure 7 Proposed Seven Node Network (DAR-80-10, 1979: 4)

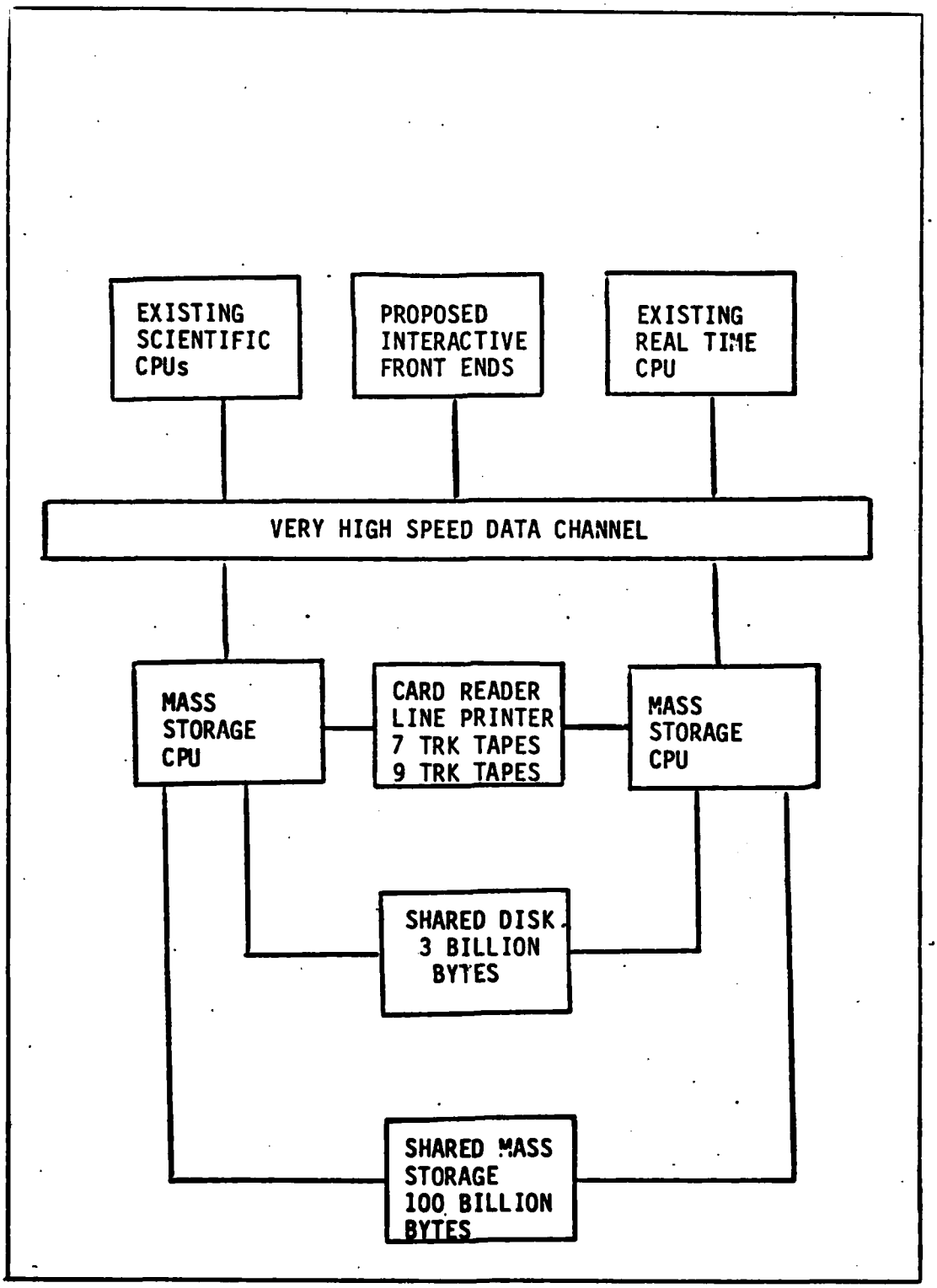


Figure 8 Proposed Mass Storage System (DAR-80-11, 1980)

personnel have decided that an on-line mass storage system also the future problem of expanded terminal use. Some of the other underlying objectives of the mass storage system include:

1) Provide a virtually unlimited growth potential in on-line storage
2) Provide a system-independent storage facility
3) Provide an ultra-reliable data storage system
4) Mesh with oher ECONET plans for upgrade

The configuration of a mass storage system can be a very integral part of a backend database computer system. How the data are stored and accessed is a critical performance factor for database queries.

Backend Database Processor

As described, the future configuration and the modular upgrade approach should easily facilitiate the addition of a backend database processor. There are several "software" backend processors available as well as "hardware" versions of a backend architecture that look very promising. The high speed communications bus should allow for low communication overhead time no matter where in the system the backend processor is placed. Also by having several interactive front ends, much of the query processing and reduction can be done quickly by the IFE before a transmission is ever made to the backend. Another advantage will be the large amount of on-line storage available with the proposed mass storage system. The

backend processor and mass storage combination should allow for fast access to very large databases. Database applications should no longer be a bottleneck in the ECONET system.

Backend Database Computer System

Concepts

The basic concept of the backend database computer is to remove all or part of the DBMS from the host computer and put it on a backend computer system (Maryanski, 1980: 3). Generally speaking, this backend computer may be a special purpose computer called a Database Computer (DBC), which is a hardware implementation of the concept, or it can be a general purpose computer (software implementation), usually a mini or micro computer. A basic backend database computer system architecture is shown in Figure 9. In this model the application programs are executed by the host computer while the backend machine controls access to the database. The term backend was selected to describe this configuration because of the obvious similarity to front-end computers. A front-end computer serves as the interface between a host computer and its external inputs while the backend serves as an interface between the host and its database (Canaday, 1974: 576).

The functions that a backend database computer system performs depend partially on the configuration of the system that it supports. It could support a single host, a

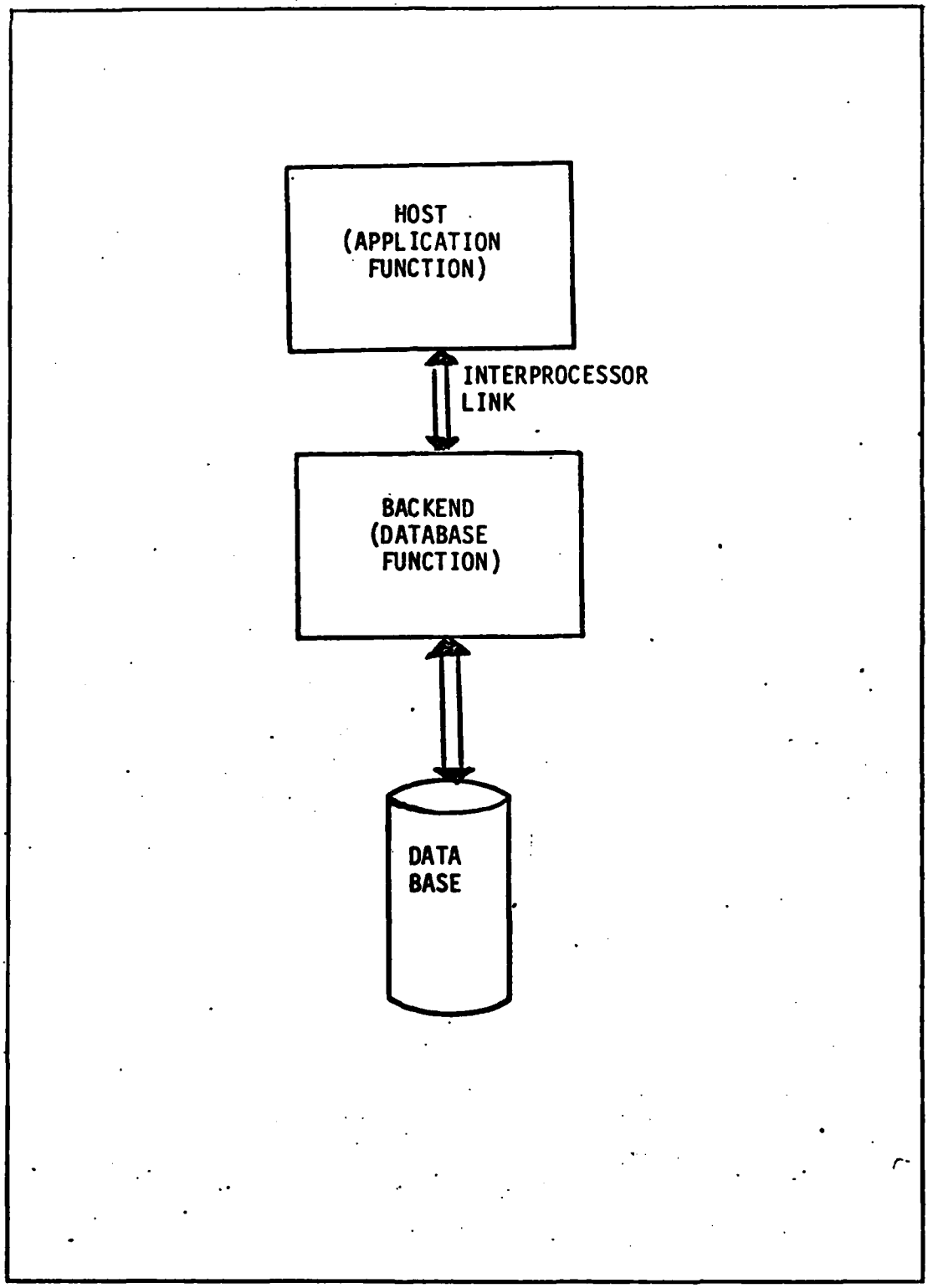


Figure 9 Backend Data Base Computer System
(Maryanski, 1980: 4)

two host, or (see Figure 10) a multiple host configuration (Canaday, 1974: 577). Another possibility is that a backend-processor could be one of several backend processors that support a single database system or multiple database systems. (Canaday 1974, 578) (see Figure 11). The configuration used also depends on whether a hardware implementation (DBC) is used or if a software implementation is used.

No matter which implementation is used, however, the primary purpose of the backend database computer system is to relieve the host of some or all of the CPU resources required for processing database queries and to have a dedicated computer for database I/O. The performance increase associated with a backend architecture comes about primarily because of the concurrent operations of the host and the backend (Maryanski, 1980: 8). When the host receives a database request it will usually only perform some minimal functions (such as checking whether access is permitted) and then pass the remainder of the database operation to the backend. Once the tasks have been given to the backend, the host's resources are freed to process other requests. The host and backend then operate concurrently on different tasks until the backend machine sends the requested database information back to the host. There is however a performance penalty incurred with the backend database machine by the introduction of the interface and communication software and hardware, and the

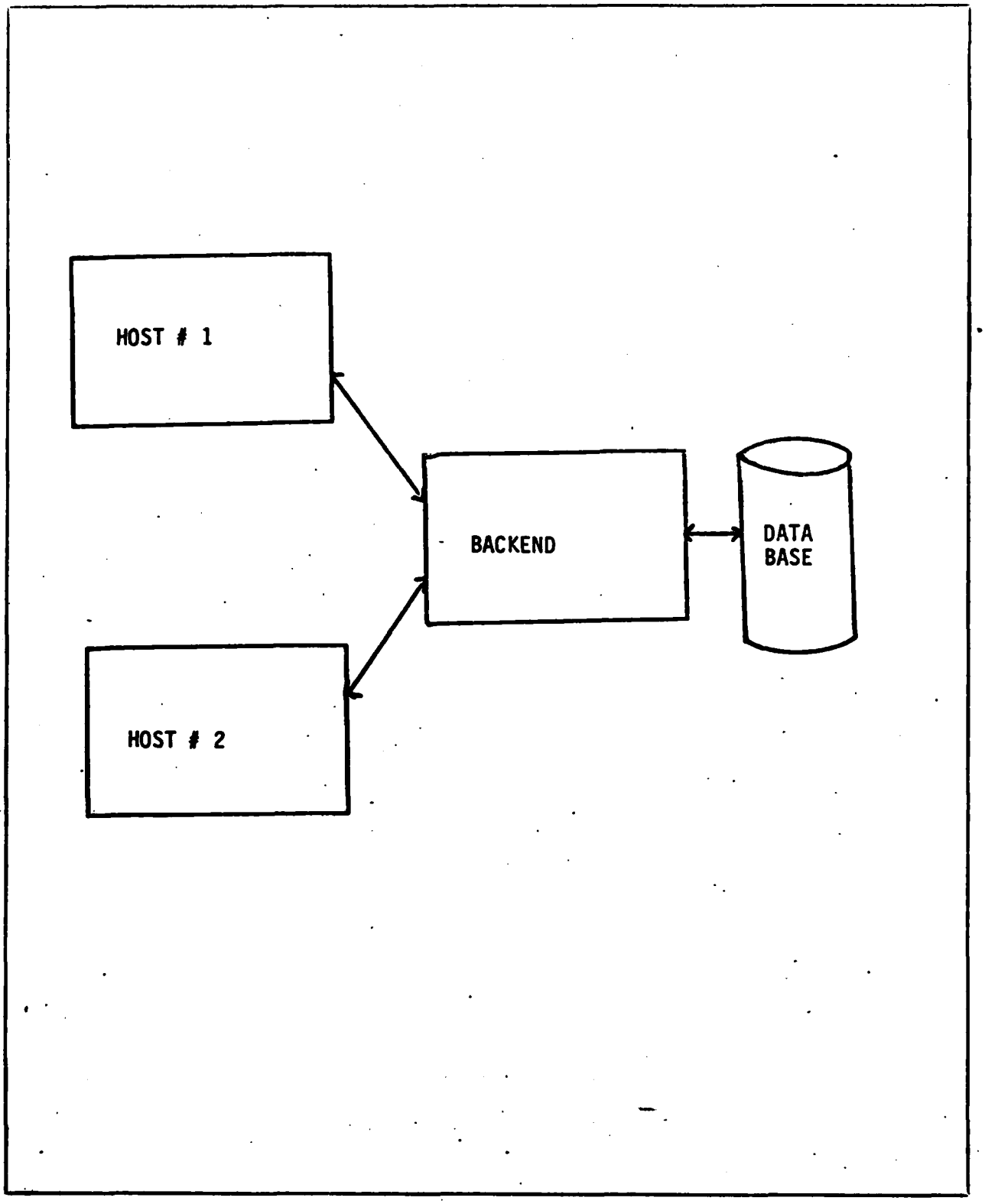


Figure 10 Multiple Host Configuration (Canaday,1974: 577)

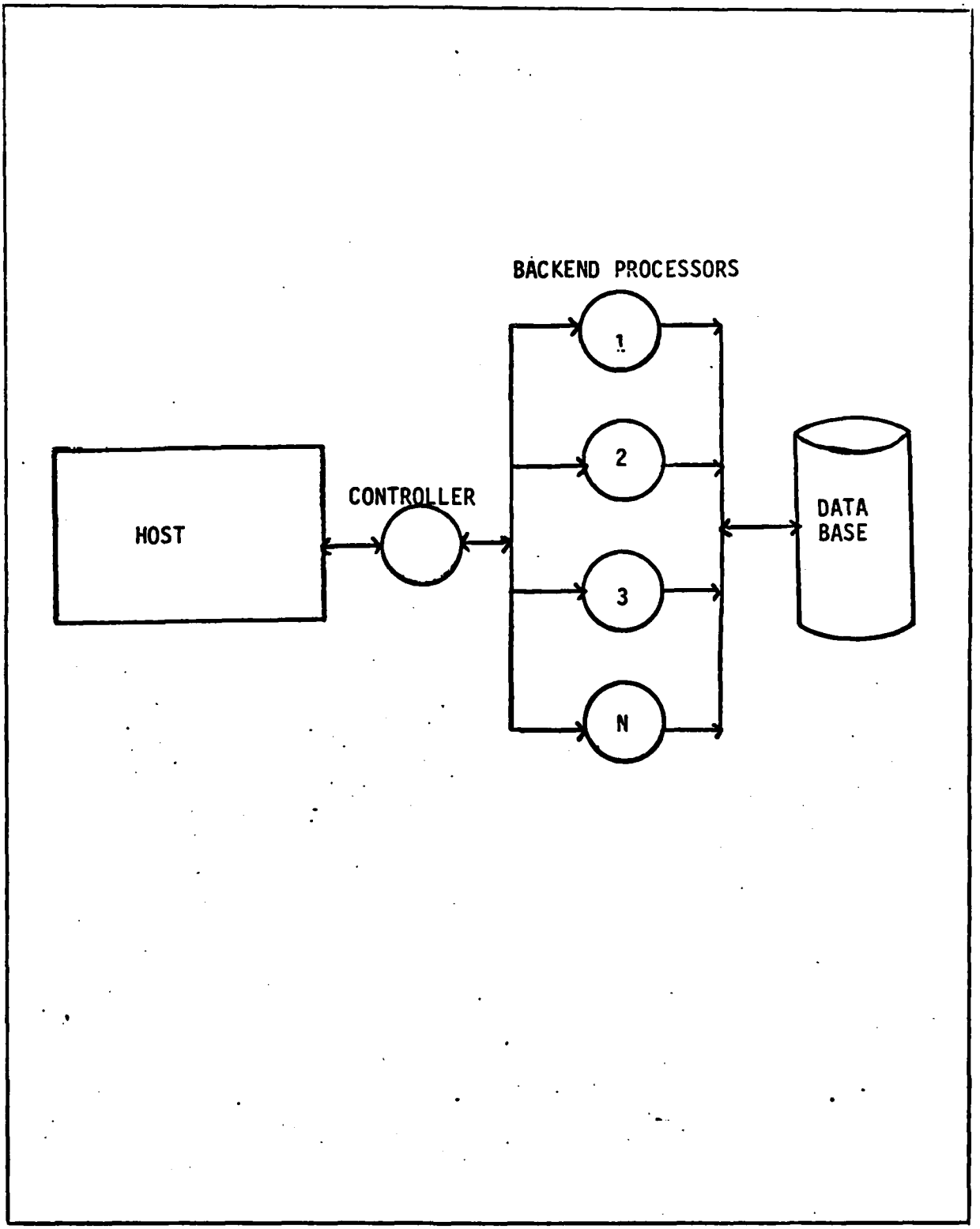


Figure 11 Multiple Processor Backend System

transmission time of the intercomputer link. In fact, the execution of a single database command will take longer on a backend configuration than on a single machine configuration because its data must pass through more modules. In order to realize the concurrency benefits, the operations of the backend processor(s) and the host processor(s) must be synchronized (Maryanski, 1980: 8). Also a certain minimum level of CPU usage must occur. The principles are similar to those surrounding multi-programming. There is a certain load at which the system is most efficient. Within certain bounds either a smaller or larger load can adversely affect performance.

In order to further facilitate the performance of a backend database machine two additional conditions (for a total of 3) must also be met: (Maryanski, 1980: 8)

1. A high speed link must be used between the host and backend.
2. A substantial demand for database access must exist.
3. The backend computer must be multi-programmed.

In reference to condition 1 Maryanski suggests that the communication link should be at least 1M baud or that shared memory be used. Simulation studies and prototypes have shown that lesser data transfer rates produce communication delays that cause bottlenecks. (Canaday 1974, 582). Also to handle several database tasks concurrently the backend database computer system must be

multiprogrammed (Canaday, 1974 and Maryanski 1976). These three minimal conditions should present no problems for most installations. The technology is available today for several different approaches to backend database computer systems.

Now that the general concepts of a backend database computer system have been described several of the potential advantages and disadvantages will be discussed.

Potential Advantages

Performance Improvement

This of course is the primary reason for the extensive investigation into backend database computer systems. Software implementations have already shown as much as a 25% increase in response time over conventional systems. (Maryanski 1980: 8) Similarily, at least one hardware approach claims an order of magnitude increase in performance (Banerjee: 1978, 376). In order to realize these gains, however, the three conditions previously mentioned must be met. If a backend configuration is to be successful, the gains due to concurrency must outweigh the losses caused by communications overhead.

Cost

Many computer facilities automatically buy a larger mainframe when the one they have becomes inadequate. The backend database computer system offers a much cheaper (and probably more effective) solution to the database saturation problem. Mini or Micro computers are generally

an order of magnitude cheaper than a larger mainframe (Heacox, 1975: 511). However, to realize this cost benefit, factors other than processor costs must be evaluated. The media used for storage of the database on the existing mainframe and the backend must be compatible. Also communication lines and interface software/hardware must be considered (Maryanski, 1980: 8). Even with these other considerations, however, the backend database computer system offers an attractive economic alternative.

Reduction of Host Workload (Fisher, 1976:294)

Moving the database operations to a backend system helps the host computer in several ways. First, the host memory allocated to the DBMS is significantly reduced. It is true that part of the freed memory may be replaced by communications software, but the amount should not nearly be as much as before. Second, the amount of memory for application programs on the host computer is reduced. Third, operating system overhead of the host is reduced, since many operating systems functions are assumed by the backend. With this reduced database workload the host is now free to process other applications.

Enhancement of Database Security and Integrity

Computers that are controlled by traditional general purpose operating systems are notoriously ill-suited for database protection. The backend computer should help remedy this problem. The backend and its database are not

directly controlled by the host's hardware and operating systems. A properly constructed backend is an autonomous filter between the host and database. This independance allows the backend to validate every attempted database request coming from the host and also to gracefully recover a database if the host fails. Although total security is almost impossible the probability of a secure system is increased when the only path to the data is through an intelligent backend (Lowenthal, 1976: 24).

Reliability

The question that arises here is whether a pair of tightly coupled machines is more reliable than one computer. The argument can be made that with two modules there is twice as much chance of a failure. However, with a backend database computer system the necessity for intermachine communication causes the machines to be able to check the information passed between them. With these checks the existence of a problem should be detected more quickly, thus reducing the time the system operates with an error (Heacox, 1975: 511-513). The earlier errors are detected the less time is spent in back up and recovery.

Canaday elaborates on this idea of dual checking by suggesting audit trails on the host and backend. Each audit trail allows rollback no matter which machine fails. He believes that two machines checking each other is much more reliable then one machine making a self inspection and easily offsets the decrease an overall system hardware

reliability due to the additional backend equipment (Canaday, 1974: 577).

Economy Through Specialization (Canaday, 1974:576)

By creating a functionally separate module that handles only database functions, several economies of specialization are realized. The backend's operating system can be tailored to serve just database processing. This means the backend computer should, for example, have good byte manipulation and have high input/output thruput. It also means that the backend does not need floating point instructions, fast multiply and divide circuitry, a large word size for high precision or a wide variety of peripherals. Economies therefore should be seen in (1) a smaller on-line system requiring less core (2) simpler programs requiring less processing time (3) smaller development costs (4) a shorter development cycle. A specialized processor with an efficient encode-decode capability can also allow the data to be compressed with encoding techniques which will reduce database storage requirements. (Nagel, 1981: 34).

Low Storage Costs (Madnick, 1977)(Hsiao, 1977)

The storage cost per byte can be reduced by pooling the storage requirements for several computers and using more economical high-volume storage devices. Also by using mass storage devices, large databases that may have been previously spread over several devices with necessary redundancies can be stored on one device managed by the

backend computer. This type of distribution of redundant data is a problem on the ECONET and therefore makes the low storage cost advantage worth investigating.

Extended Usefulness of Current Machines

By off-loading the database management function from a saturated host computer on to a backend database computer the effective life of the host may be extended for several years. Many of today's DBMS's do not effectively operate on computers manufactured in the 1960's and 1970's. The CDC Cyber 176 at Eglin, for example, was designed as a "number cruncher", not an interactive database processor. By allowing the older machines to specialize in the functions they were originally designed for their usefulness should be extended; thereby delaying upgrade.

Modularity

The principle of modular design is generally applied in the development of hardware and software in order to provide well defined, reliable, and correct computers and programs. A backend database machine is an example of a modular design at the computer system level. This modular approach also fits in nicely with the modular upgrade approach taken with the ECONET. This separation of function allows multiple hosts to interface with a single database or distributed database networks (Maryanski, 1980: 9). This is a much simpler task than modifying the entire DBMS or developing methods to allow two independent

DBMS's to concurrently access the same database.

Disadvantages

Cost of a Second Machine

Even if the backend database machine is cheaper than a host computer upgrade, there is still a cash outlay for the second machine. Before spending any money the viability of a backend system should be studied and other alternatives explored. Aside from the question of actual purchase or leasing costs is the likelihood that the backend computer would be manufactured by a different vendor than the host. This can duplicate the problems associated with contracts, maintenance, operator training and systems programming support.

Unbalanced Resources (Canaday, 1974: 578)

Once you have dedicated the database management functions of a host machine to a backend machine there is a possibility of uneven load between the two processors. With a backend configuration you are less flexible in your ability to balance loads as requirements change. If the host is busy while the backend sits nearly idle (i.e. few database requests) then you are wasting expensive processor time. In this situation it may be questionable if a backend database machine was even warranted. If it is the backend processor that is overworked, one may consider upgrading it or possibly using a multi-processor backend configuration. Another way to increase the workload of an under used processor is by sharing it with another user.

The unbalanced condition could be permanent, random or even oscillate between the backend and host. The exact nature of the unbalance could easily impact the decision of how to solve the problems of unbalanced resources.

Response Time Overhead (Canaday, 1974: 578)

New overhead times introduced by adding a backend database machine include (1) transmission time of the command to the backend (2) transmission time of the results back to the host (3) task queueing delays related to these transmissions and (4) possible conversion overheads associated with incompatible word lengths, character sets and data formats. The response time associated with 1 and 2 can be greatly reduced if a high speed bus is used or a shared memory scheme is used as in the MADMAN machine (Hutinson 1978: 85). The times associated with 3 and 4 must be dealt with in a system by system basis.

Two Approaches

Having given some of the general concepts and advantages and disadvantages of backend database computer systems, two different approaches will now be presented. First, a general discussion of each of the approaches will be given followed by descriptions of notable prototypes in each area. One should not view the two approaches as one being "better" than the other, but rather they should be viewed as each effort being a building block for the next.

The Software Approach to a Backend Database Machine (Maryanski, 1980: 6)

Figure 12 shows the software organization of a backend DBMS. Here the functional modules of the single machine DBMS have been distributed between the host and backend machines. Because they are separated, communication interface modules must be added to both computers.

The communication system must provide the facilities for the transmission of commands, data, and status information between the host and backend. The control of the physical links, proper utilization of line protocol, transmission error detection and correction, processor and task synchronization, and management of message buffers are among the responsibilities of the communication system As previously mentioned, the link between processors should be at least 1M baud to avoid excessive response time overhead (Maryanski, 1976). The interface routines are the processes that exchange information via the communication system. The host interface (HINT) is the process that is called by the application program when access to the database is desired. More specifically, whenever an application program encounters a data manipulation language (DML) command the HINT is called. The HINT formats the database request according to the specifications of the communication system and then transmits the request as a message to the backend interface (BINT) via the communication system. When the database

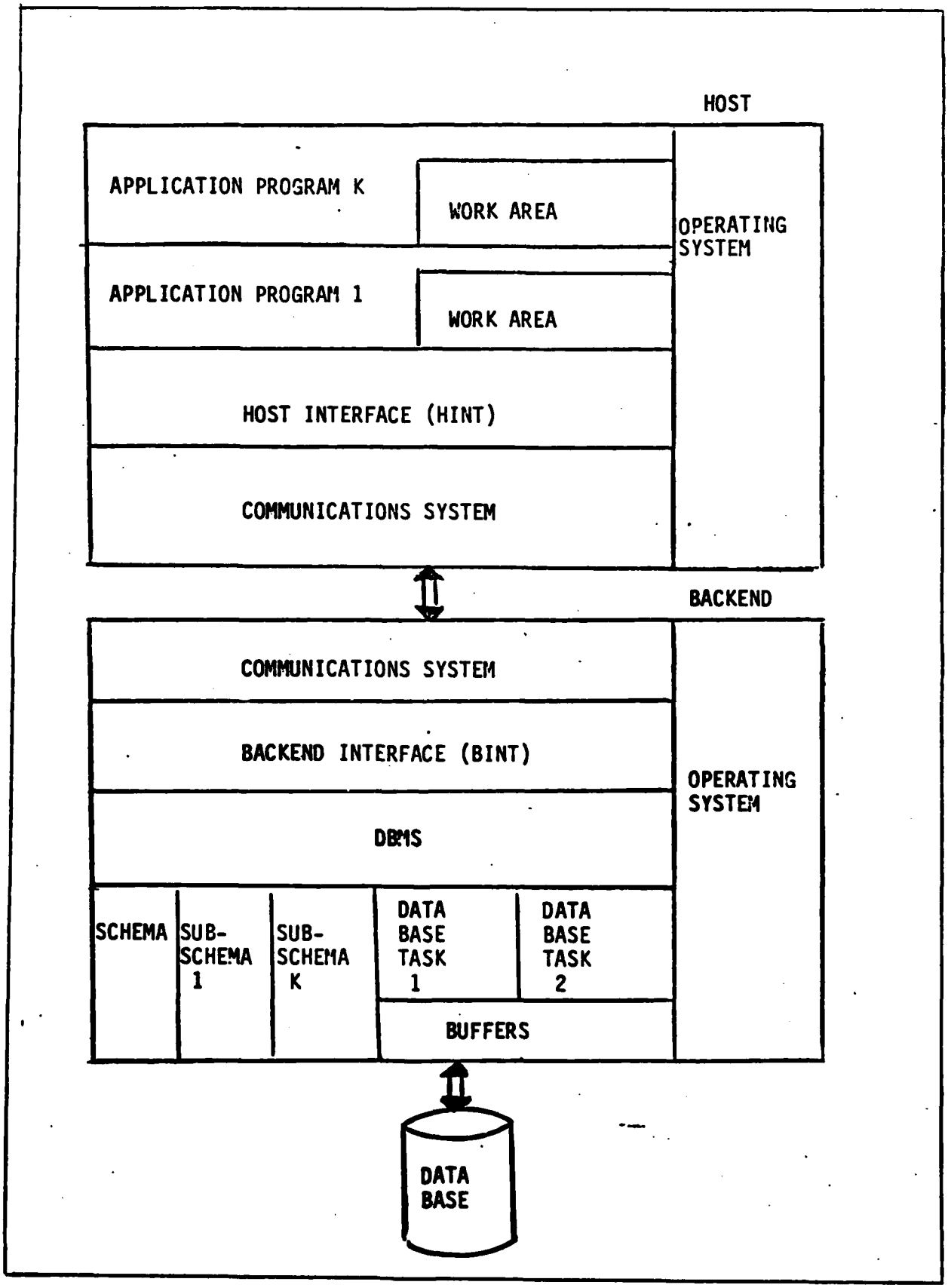


Figure 12 Software Organization of a Backend DBS System (Maryanski, 1980: 6)

operation is completed by the backend the results are sent back to the host in a similar fashion. Note that Figure 12 shows a number of database tasks located on the backend machine. This shows the multiprogramming that is necessary if the backend is to improve performance. The database task is like an application program on the host. The entire operational sequence is now presented (Maryanski 1980: 7):

(1) Origination and Validation - The application program issues a request to the HINT indicating the operation required and the data to be operated upon.

(2) Host-to-Backend Communication - A message containing the database request is sent from HINT to BINT via the communication system.

(3) Data Access - The BINT passes the request to the proper database task, which calls the Software DBMS. The DBMS checks its buffers to determine if the request can be satisfied with currently available data. If it is available the resultant data is passed through the database task to the BINT. If the data is not immediately available the DBMS requests the backend's operating system to transfer the data from secondary storage to the backend's buffers and then to the BINT through the database task.

(4) Backend-to-Host Communication - A message containing the data and status information produced as a result of the operation is transferred from BINT to HINT via the communication system.

(5) Completion - The HINT places the result into the work area, where it may be acessed by the application program.

XDMS

The first backend database machine prototype was developed at Bell Laboratories by Canday et al. It was called XDMS. The principal motivating factors in the development of this first backend machine were interests in the CODASYL data model and in minicomputer applications. The configuration implemented is depicted in Figure 13 (Maryanski, 1980: 10). The DMS-1100 DBMS (which is a CODASYL - based package) originally resided on the UNIVAC 1108, but was moved to the backend machine, a META-4. A personnel application system was moved to the backend for testing purposes. Keeping in mind that the primary purpose of XDMS was to prove the concept of the backend computer, the project was a success. It was this prototype, however, that discovered that the communication link must be at least 1 M baud. The original design used a 2 K baud link which proved to be the system's bottleneck. Other theories about the principals of backend computer did however, prove correct. As the XDMS backend became busier more concurrent operation occurred and the system's performance increased. The XDMS project is a landmark in the database management area, since it introduced an alternative database architecture and spawned an enormous amount of follow-on research.

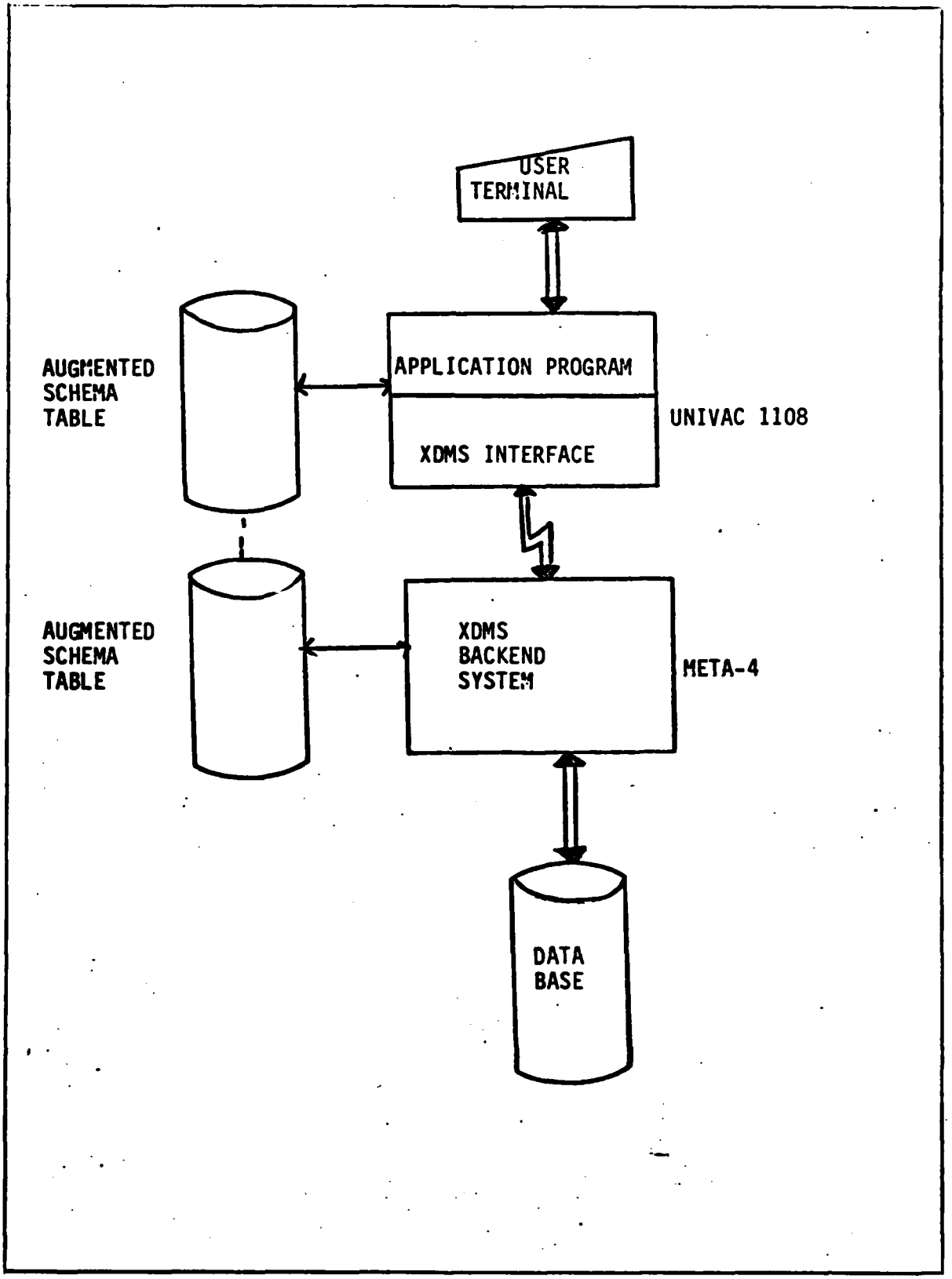


Figure 13 XDMS Hardware Configuration (Maryanski, 1980: 10)

MADMAN (Hutinson, 1978) (Maryanski, 1980)

The bottleneck created by slow communiction lines in the XDMS prototype prompted development of the MADMAN prototype (Multi-Access Data Manager). By using compatible processor's and a shared memory connection as shown in Figure 14, MADMAN permits high-speed interprocessor communication with minimal overhead. The disadvantage of this scheme is that the host and backend computers can not be physically separated by long distances. The MADMAN is designed to be an intelligent I/O device with direct memory access (DMA) to the host computer. The general sequence of events is as follows:

1) The host tells the backend to start processing and gives it a pointer to a list of command blocks for request to be processed.
2) Whenever the backend has completed a request it interrupts the host to signal the completion of a request and also indicate which request is done.
3) Other requests (if any) are continued.
4) The host adds command blocks to the command list as it receives requests from application programs.
5) The backend takes requests off of the command list when it is ready to process another request.
6) The host tests to see if the request is the last on the list. If not, then the host must restart the processing of the command list.

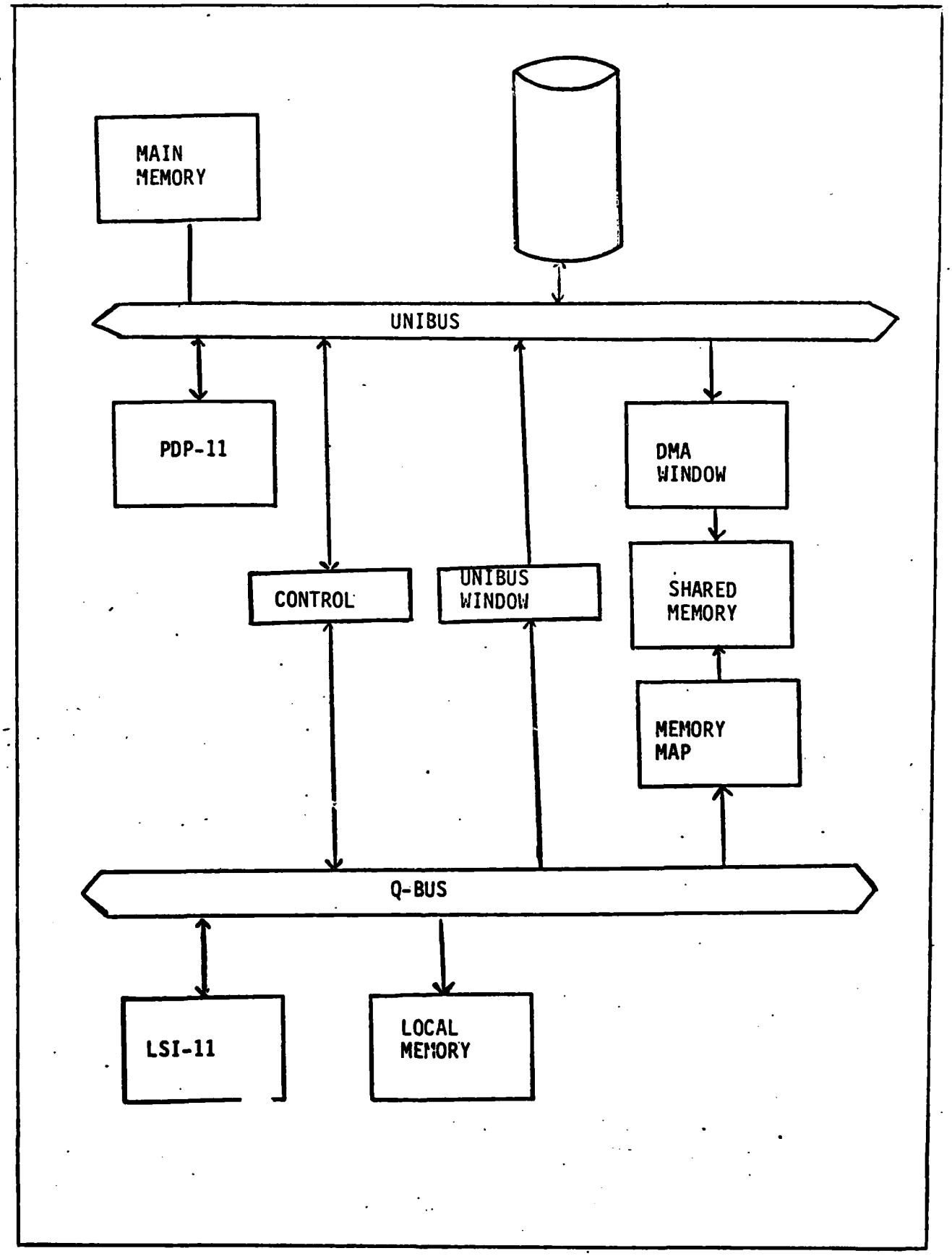


Figure 14 MADMAN (Hutinson, 1978: 87)

Because of I/O delays, the requests may be completed in an order different than they were originally queued to the backend. This causes no problem because application programs are allowed only one request to the DBMS at a time.

MADMAN differs significantly from other backend systems. The most important differentiation is the direct connection of host and backend via shared memory and bus linkages. Only the CPU portion of database activities are actually off-loaded to the backend machine. MADMAN is currently being used in a number of General Electric factories along with an associated transaction processor. It has shown improvement over some of the earlier prototypes such as XDMS, especially in the area of communications overhead. As mentioned before, however, the host and backend must be located close together because of the shared memory concept.

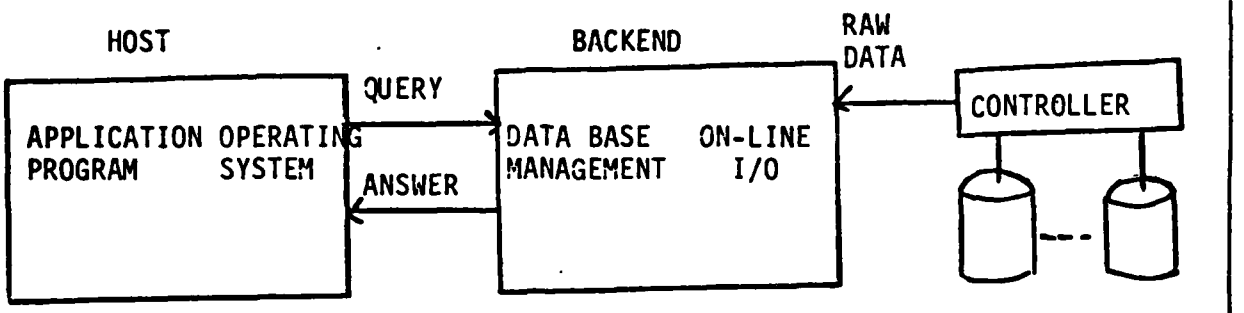
The Hardware Approach to a Backend Database Computer System

More recent research on backend computers has centered around a generalized hardware solution. As processors and microprocessors become cheaper, specialized processors are being used more often. Hardware has begun to replace many functions that were previously done completely by software. Hardware solutions have shown to be much faster, more reliable and in the long run cheaper because of the increasing cost of software development and maintenance.

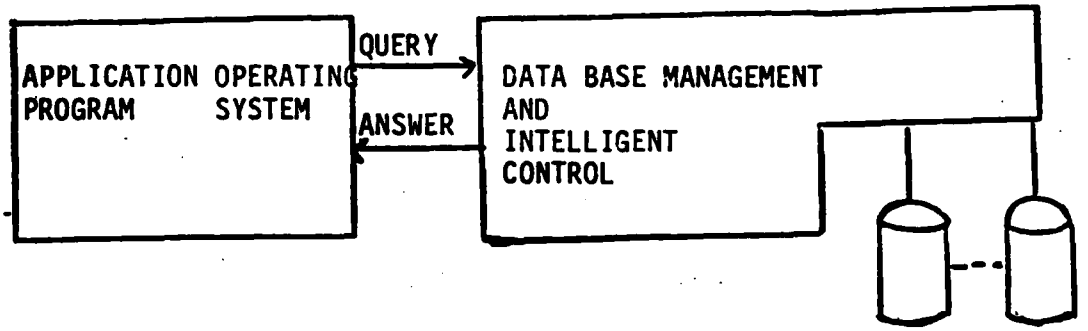
For these and other reasons a hardware solution to the backend database computer system has received much attention from researchers.

According to David Hsiao (Hsiao, 1981:5) the software backend versions do not solve the 90-10 rule of contemporary database systems. Basically, this rule says that for very large databases, in order to find the relevent data, nine times as much additional irrelevant data will have to be brought into the main memory for processing. That is, only 10% of all the data processed will be relevant to the request. The software backend design only delays the 90-10 rule. This is better than the traditional system however, but the hardware solution actually overcomes the 90-10 rule (see Figure 15). The disadvantage of software solutions is that they are conventional computers performing data management tasks by software means. The difficulties of name mapping and data updating is still an integrated part of these machines. The hardware backend, however, is almost completely without software and can support a very large on-line database. The design has a large content-addressable storage due to a partitioned content-addressable memory (PCAM). The hardware approach, therefore does not require staging of data between levels of memories at various speeds. There are three key concepts that are designed into the hardware that account for its greatly improved performance:

1) Associative memory



SOFTWARE BACKEND APPROACH - DEFERS 90-10 RULE



HARDWARE BACKEND APPROACH - ELIMINATES 90-10 RULE

Figure 15 The 90-10 Rule (Hsiao, 1981: 5)

2) Parallel Processing

3) Pipelining

Associative Memory (Baer, 1980:262)

Associative or "content-addressable" memories have the capability of addressing items by content rather than by location. Rather than answering a query "What is the content of location 12345"?, an associative memory asks "Is there a location containing item "Joe"?" To do this search efficiently all locations must be searched in parallel. Because of this, each bit of the associative memory requires logic. This additional logic, until recently has made larger scale associative memories expensive. However, with cheaper processors becoming available, some parallel processing associative memory architectures have been developed. The database computer at the Ohio State University is one example. This system proposes specialized processors and modified moving head disks to attain very large content-addressable blocks using tracks-in-parallel readout (Kerr, 1979: 78-79).

Parallel Processing

The use of several special processors working in parallel is what makes the associative memory concept feasible. They consist of many processors capable of performing synchronously the same operation under the supervision of a common control unit. The DBC has several modules which use parallel processing.

Pipelining

Pipelining means to decompose a process into a series of sequential subprocesses. Each of these subprocesses is then executed on a dedicated facility called a stage or station (Baer 1980: 500). A controller is generally used to ensure that all transactions move smoothly through the system, so that all stations can be executing specific functions for different commands at any instant of time. The pipelining concept can be used at several levels depending on the degree of subdivision of processes. Fonden for example, (Fonden, 1981: 37), applied the concept with relational database queries to allow pipelining to process tuples that can be manipulated independently and in parallel.

The Database Computer (Banerjee, 1978)

Jayanta Banerjee, David K. Hsiao and others at the Ohio State University have designed a hardware backend computer called the Database Computer (DBC). Their research and simulations show that their design can perform up to 160 times faster than current software database management systems. Hsiao mentions four problems faced by conventional software DBMS's that are solved by their design. The problems include name-mapping, performance bottlenecks, data security overhead and an add on approach to data security.

Problems with Conventional Systems

The complexity of conventional DBMS software is due, in large part to the requirements for name mapping operations. Name mapping operations convert symbolic data names (called a query) into storage addresses which identify where the data named by the query can be found. Because in most systems the language used to name data is far more powerful than the addressing scheme used by the hardware, complex name-mapping algorithms must be used. In addition, name-mapping algorithms must be highly optimized to perform well. Therefore in a conventional software DBMS the supporting hardware design can greatly determine the complexity of the software.

Performance bottlenecks, according to Hsiao, are another problem of the conventional DBMS. Database system software normally consists of several distinct functional parts which perform specific tasks. There are separate modules for query parsing, directory access, directory processing, and data retrieval. To have high system throughput, these modules must have diverse performance capabilities. However, since the hardware that these modules reside on is the same, this diverse performance is not generally available. Once again the hardware is constraining the system's software, as with the mapping problem, and the result is performance bottlenecks.

Data Security overhead is another performance hindrance of conventional systems. Authentication operations can require several name-mapping operations.

There are really no DBMS's available today that can call themselves totally secure. Generally speaking, with conventional DBMS's, the more secure the system the worse the overhead to support that security. Also most present security systems are not built into the DBMS, but are added on. This kind of design philosophy opens the way for poor reliability and for performance difficulties. The security system should be an integral part of the DBMS.

The Problem Solving Concepts of the DBC

Hsiao presents several design considerations that he claims will solve the previously mentioned problems associated with a conventional DBMS. By keeping in mind that his overall objective is to increase performance as much as posible by replacing software functions with hardware, we can better understand his approach. Hsiao's problem solving concepts advocate the use of:

1) Partitioned content addressable memory (PCAM)
2) Structure memory and mass memory
3) Area pointers
4) Functional specialization
5) Look aside buffering
6) An Integrated Data Security Mechanism
7) Clustering Techniques
8) Emerging and existing technology

PCAM

PCAM helps solve the name-mapping problem. The use of

hardware content addressing greatly reduces the need for name-mapping structures. Content addressable memory eliminates the need to know the actual location of a data item. This capability allows data items to be moved without any need to modify name-mapping structures. Since a very large content addressable memory is not yet feasible, Hsiao's design partitions the memory (hence PCAM) with a specialized processor handling each partition. These multiple processors allow parallel content addressable searches and thereby greatly decrease access times (as compared to conventional systems).

Structure Memory and Mass Memory

Since PCAM's do consist of several partitions there will be some name-mapping necessary even in Hsiao's DBC. This function is handled by the structure memory. A DBC therefore has two memories, a structure memory to map the partition and a mass memory (the sum of the PCAM's) to store the majority of the data. The mass memory contains only update invariant data structures. The data structures in a conventional system are not update invariant; they must be modified whenever the location of data changes. The structure memory contains all of the non update invariant name-mapping information necessary to locate data in the mass memory. When a database access is made the system first accesses the structure memory (which is also PCAM), obtains mapping information (to a block of mass memory) processes it and then accesses the mass memory.

Area Pointers

Area pointers are used to simplify the name-mapping data structures that are still required by the structure memory. Area pointers point to a PCAM partition and are stored in and managed by the structure memory.

Functional Specialization

In a functionally specialized system, the components are individually designed to be optimally adopted for their function. Hsiao's DBC contains several physically separate components that are functionally specialized. This design also allows the use of pipelining techniques which permits subprocesses to be executed simultaneously with other subprocesses on the different functional hardware components. The seven components and their interrelationships are shown in Figure 16. The components are the keyword transformation unit (KXU), the structure memory (SM), the mass memory (MM), the structure memory information processor (SMIP), the indextransformation unit (IXU), the database command and control processor (DBCCP) and the security filter processor (SFP). The structure memory and mass memory have already been discussed, the other functional components will now also be discussed.

Keyword Transformation Unit and Index Translation Unit

The KXU and IXU are mainly for the efficient realization of the DBC.

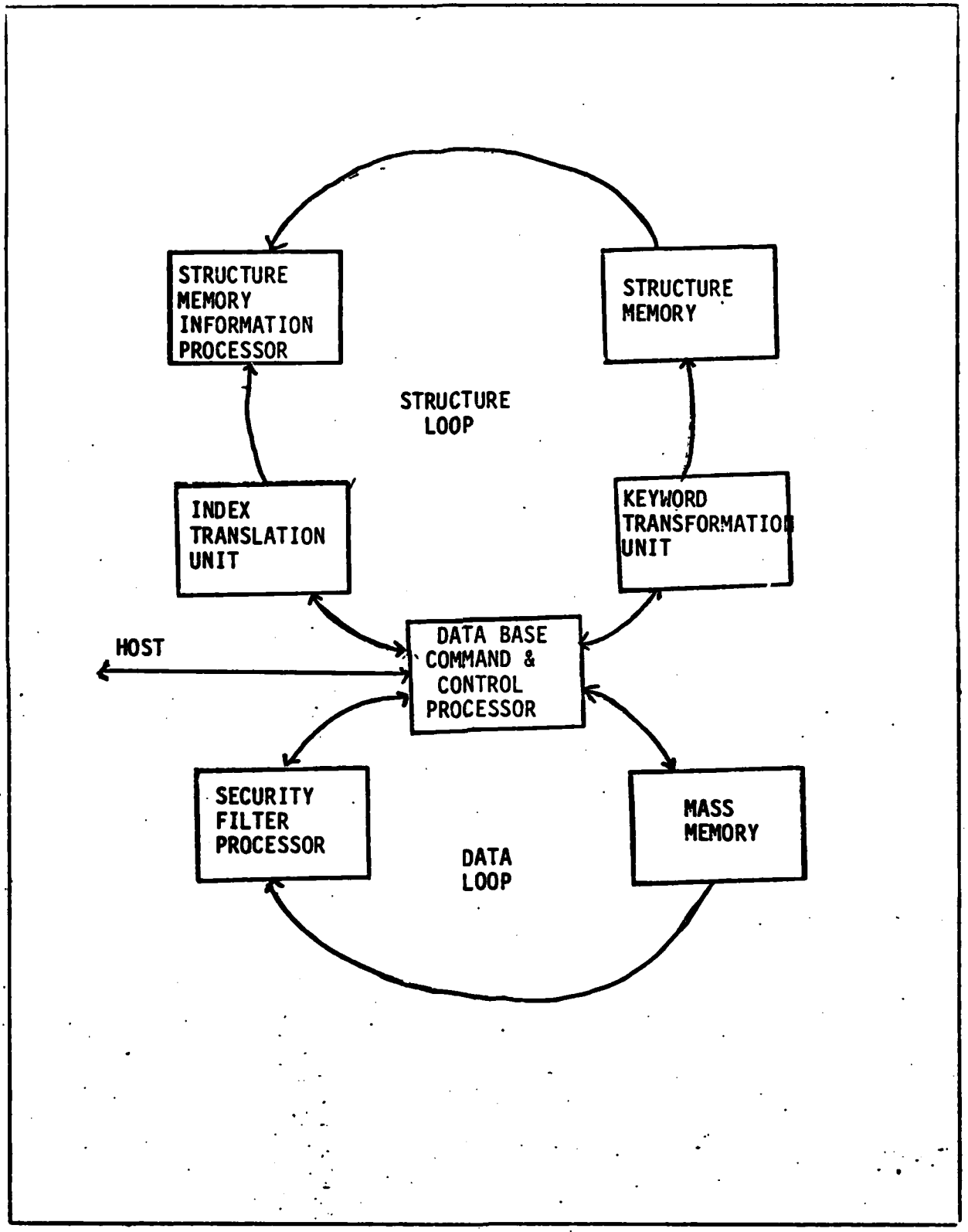


Figure 16 The DBC System Architecture (Banerjee, 1979: 415)

Structure Memory Information Processor

The SMIP is basically a set operation processor. Set manipulation operations are performed by maintaining an intermediate set in the SMIP while the argument sets which modify it are passed through the SMIP. The SMIP, like mass memory, also uses a PCAM organization. It achieves a very high speed by using many processor memory pairs to execute set operations in parallel.

Database Command and Control Processor

The DBCCP is the brain of the DBC. It regulates the operation of the entire system. Its basic functions include: receiving commands from the host (a front-end), executing those commands by properly controlling the various parts of the DBC, and sending a response back to the host. It is the DBCCP's responsibility to move commands smoothly through the system, forming a pipeline so that different functions can be executing simultaneously. The DBCCP also enables clustering, a concept which will be discussed later in this section.

Security Filter Processor

The DBC has a built in security processor (SFP) that along with the DBCCP jointly maintains the database capabilities for the users on the system at any point in time. The backend concept in general enhances security simply by allowing access to the database through the backend. The DBC even further enhances this inherent

security by using a separate security processor. In order for the SFP and DBCCP to perform their security function, however, the proper access authorization must be provided by the program execution system (PES). The PES according to Hsiao, is any system that talks to the DBC, for example the host or some type of front-end. A table is kept for each user with the database capabilities for each actual file. A table entry would have the form:

$$(F, (Q_1,A_1), (Q_2,A_2),\ldots,(Q_m, A_m)$$

where each Q_i is a query, each A_i is an access set, and the set of couples is a database capability.

Look-Aside Buffering

As previously mentioned the structure memory and structure memory information processor work together to alleviate much of the name-mapping problem associated with conventional DBMS's. There is, however, a small amount of name-mapping necessary on the DBC because of the necessity to partition mass memory. The changes induced by update operations will be small because the partitions are large, but even these changes in the structure memory will require time. To reduce this time a fast look-aside buffer is used. The update changes are recorded in the buffer and are referred to by all subsequent commands until the changes can be permanently recorded in the structure memory during slack periods of operation.

An Integraded Data Security Mechanism

The DBC provides security through two protection mechanisms. The first is based on the security atom concept and requires action in the part of the database creator. Enforcement is accomplished with the DBCCP. The second mechanism allows the database creator flexibility in the way he can specify security-related information. This is where the SFP is used. When designing the DBC, the security system was actually designed first and the remainder of the system built around it.

Clustering Techniques

The DBC uses a powerful clustering technique to allow for even faster data access. The database creator, knowing the relationship of the data, can specify clusters of records based on the properties of the data. To support the record clustering operation a cluster table keeps track of the clusters and partitions of mass memory to which those records belong that have the same clustering keywords. Generally most clusters should fit within a partition so access times will be enhanced. However, if the cluster does become too big for one partition the cluster table allows for spill over to additional partitions.

Emerging and Existing Technology

There are many emerging technologies which the DBC will take advantage of. In fact, as described here, the

DBC can not be realized with just today's technology. The DBC will use low-cost microprocessors coupled with low-cost moving-head disks with tracks-in-parallel read-out to allow high-volume processing with content-addressaable capabilities. Also low-cost random access memory will allow the widespread use of very large data buffers and independent, functionally specialized memories throughout the system. The availability of inexpensive and powerful microprocessors will allow a very effective PCAM architecture. The PCAM will consist of a small number of very large content-addressable blocks and is realized by a larger number of microprocessor-memory pairs. This PCAM organization could support a great variety of keyword predicates. This is because all keywords of a given attribute could probably be stored in one partition, therefore allowing any predicate to be applied to all keywords of that attribute with one access.

Summary

Now that the reader has a basic idea of the concepts surrounding the backend database computer, a better understanding of how these concepts can help the ECONET should be possible. Chapter three will present an analysis of the ECONET and how a backend computer could be included in Eglin's modular upgrade approach.

III. SYSTEMS ANALYSIS

Introduction

This chapter covers the systems analysis done of the ECONET at Eglin AFB. The analysis was approached with the objective of finding information to accurately simulate the performance of a backend database processor when added to the ECONET. The backend would be added to the configuration shown in Figure 4 (Chapter 2). Several techniques were used to accumulate the necessary data for a simulation including: workload studies, interviews, investigation of vendor supplied performance characteristics, and the review of historical records and management tools.

The workload studies were conducted by Eglin personnel to help support their decisions for systems upgrade. The studies describe past work levels and system usage and also project future levels. Most of the data useful to this thesis from the workload studies, however, are of a general nature. For example, the management information systems (MIS) portion of the ECONET workload was found to be 14%. This kind of information may give general trends about applications, but it gives little "hard" data for simulation purposes.

Informal interviews were conducted with members of the MIS branch and the Systems branch. The information gained here was useful both for providing simulation data and for leads to other areas of investigation including locally

available vendor documentation. Several historical documents and automated management tools were also located as a result of various interviews.

An important point to remember is that the systems analysis was done to provide a baseline for predictions to be made about the system's performance in 1986 (projected installation date of a backend DBMS). By that time workloads and systems architecture will have changed. The analysis was done with the intention of determining "where the ECONET is now" and also "where it will be going" in the future. Undoubtedly in such a situation several assumptions must be made when predicting parametric models and other information for input to a design and simulation of a computer system. These assumptions will be discussed later in this chapter and in the chapter on the system's simulation (chapter 5).

Analysis of the Current System and its Environment

The current system is transitioning between the configurations shown in Figures 1 and 2. The inter-computer bus has been installed and two interactive front ends are in the process of being installed and tested on the unclassified system. Workload survey data and data from interviews however, were gathered from the configurations shown in Figure 1. This means that the data are from the most current operational system.

Figure 17 and 18 summarize some of the key environmental data about the ECONET. A general discussion

1) TOTAL USERS: Over 700 + 20 Contractors

2) TOTAL TERMINALS: 150

3) TOTAL ACTIVE PROGRAMS: Over 900

4) CURRENT WORKLOAD PROFILE BY TYPE:

	Percentage System Resource Usage
a. Real Time	3%
Interactive	21%
Batch	76%
b. Unclassified	89%
Classified	11%
c. Scientific	86%
Management Information Systems	14%

Figure 17 Current Key Environmental Data for the ECONET (DAR 80-10,1979)

1) TOTAL MIS (SYSTEM 2000) DATABASES: Over 200

2) QUALIFICATIONS PER QUERY: 75% have three qualifiers or less

3) SUMMARY OF SYSTEM 2000 USAGE DATA:

DATE	CPU HRS	I/O HRS	TURNAROUND(TOTAL IN MIN.)	#JOBS
10/1/81 - 3/31/82	159.36	816.63	1,059,905	16,949
4/1/81 - 9/30/81	116.20	631.24	1,028,077	18,718
4/1/80 - 9/30/80	211.19	283.90	1,476,688	15,508
4/1/79 - 9/30/79	214.11	168.97	819,376	13,875
1/10/78 - 3/31/79	81.31	62.73	338,357	5,321

Figure 18 Key Management Information Systems Data for ECONET (Eglin Management Control Documents)

about the data in Figure 17 and 18 will now be presented.

From Figure 17 we see that real time user's total share of the workload is small. This small share, however, places large demands on the CPU and memory of the computer during mission peak periods. The system's software is, however, specially designed to run multiple real time jobs with batch running in the "background", thus achieving high utilization of resources.

Batch work is still the primary method of running jobs. Most batch work, however, must wait until second and third shift to run because of the heavy daytime terminal activity. Batch jobs receive a lower priority than both real time and interactive jobs. Because of this low priority, the productivity of many of the batch users is adversely affected.

A large portion (86%) of the workload is scientific. The scientific applications consist mainly of data reduction of test range data and simulations of weapon systems. The MIS applications support several users from various organizations. For this reason no real generalizations about database structures or the nature of the data used in database management applications can be made. A generalization about database transactions, however, can be made. Most database transactions (made using System 2000 DBMS) require only 2 or 3 qualifiers. There are exceptions, however, such as some transactions run by the Accounting and Finance Office which have

complicated updates and retrievals.

As can be seen from Figure 18, MIS usage has steadily increased during the past 3 - 4 years. What is not shown in Figure 18, however, is the fact that most of the database jobs are run in the batch mode. This is because of the high use of the system by scientific applications. Many of the database jobs are run on a regular basis (weekly/monthly) and create large volumes of printed output. Data from interviews indicate that much of the data printed are not used. It seems obvious that if competition for computer resources were not so great during normal workhours, users could simply obtain a terminal only when necessary and make database transactions that only produce the desired results. For example, rather than scanning a "canned" report for certain data, a user could run a database query and print out only the data he or she wants, or even simply view the data on the CRT.

The present system of overnight batch jobs does not really utilize the full powers of a database management system. By putting the database functions on a separate backend database processor, the competition between scientific and MIS Applications should be greatly reduced. This would enable the DBMS to function in its primary role, as a rapid response management tool. Volumes of paper listings could be eliminated and users could get answers in seconds rather than waiting for batch jobs to run overnight or waiting for someone to scan through cumbersome listings. The introduction of a backend system should therefore

greatly increase the usage of the DBMS, because of its predicted faster response and its separation from scientific applications, it should also greatly increase the productivity of its users and allow them to use the system to provide quick answers to important questions.

Prediction of the Future System and Its Environment

Anytime one is dealing with predicting future environments and requirements they must be careful to clearly state their assumptions and explain how they arrived at their conclusions. Often a wide range of parameters are established because of the uncertainty of predicted values. The values presented in this section are estimates made by one person from his interpretation of the data available. The assumptions used in determining the parametric models (presented later in this section) include:

1) Interactive DBMS usage will substantially increase with the introduction of a faster DBMS and batch jobs will substantially be reduced.

2) I/O per interactive job during peak hours will decrease because a greater number of small interactive jobs will be required rather than large batch jobs

3) CPU usage per interactive job will decrease because of the reduced amount of data being processed

4) 10% of all interactive jobs require a cartridge retrieval from MSS.

5) There will be no time delay because of mass

storage staging contentions. This is to simplify the simulation of the Mass Storage System.

6) A maximum of one cartridge retrieval will be necessary per job. This is also a simplification for simulation purposes.

7) All large (I/O bound) non-turnaround time critical batch jobs will still be run during non peak hours

8) Peak usage hours will continue to be 0900 - 1700

9) The software interface between Digital Equipment Corporation (DEC) and Mass Storage hardware can be completed without seriously degrading the system's performance.

10) Software interfaces between DEC and IBM equipment will cause minimal time delays (100 MS). Admittedly this time may not be minimal when the system is actually running.

11) The performance capabilities supplied by the IBM and DEC vendors is reasonably accurate.

12) The majority of the software interface between DEC and Mass Storage System can be handled at the VAX 11/780 frontends (IFE's), thereby distributing the interface requirements over all IFE's.

13) The workload study projections made by Eglin personnel are reasonably accurate.

14) The complexity of database transactions will not change. More complex queries however will be modeled to evaluate the system's sensitivity to query complexity.

(i.e. 75% of all transactions will contain three qualifiers or less).

15) Messages between the high speed bus and VAX unibus will be buffered by hardware considered to be part of the high speed bus. This is necessary because of the difference in transmission rates of the high speed bus (5 mbytes) and the VAX unibus (2 mbytes)

Parametric Model Development

Based on the previously stated assumptions, the current systems analysis information obtained at Eglin AFB, and vendor supplied performance characteristics, several parametric models for simulation input were developed.

Job Interarrival Rates

This model gives the arrival rates of DBMS jobs to be processed by the ECONET. All jobs simulated will be retrievals. Appendix A shows how the job interarrival rate model was obtained. This model is a baseline. Sensitivity analysis will include substantial modifications to this and other parametric models.

Job Input/Output and CPU Requirements

Interactive (Peak Hour Jobs)

These models assign CPU and I/O requirements for interactive jobs generated by the job interarrival model. Because fewer large jobs will be necessary less I/O will be required per job (see assumption 2). CPU requirements will also be reduced from current levels (1982) (see assumption

3). Apendix A shows how the I/O and CPU models were obtained.

Batch (Non Peak Hour Jobs)

See Appendix A.

Initial Message Size

This model provides the initial message size of the queries submitted. There are 50 bytes required for header information as stated by Hsiao in the MDBS design. Because 75% of the queries have three qualifiers or less, most messages will be relatively small. Assuming one qualification takes on the average 100 bytes then 75% of all initial messages should be 350 bytes or less. A triagonal distribution will be used with the parameters:

low = 150 bytes

mode = 350 bytes

high = 550 bytes

Admittedly this model is an extremely rough estimation. It is important to remember however that this is only a baseline model. Changes will be made to the model to test the sensitivity of performance criteria to message sizes.

Mass Storage Hardware Retrieval Time

With an IBM like Mass Storage System (MSS) the data resides on either a direct access storage device or is readily available on a cartridge. If the data are on a cartridge, staging (the movement of data from a cartridge to a disk) takes from 10 to 20 seconds for a cylinder of

data, not counting contention-related waits in the Mass Storage System (Misra, 1981: 346). If the data are already on a disk the retrieval time is the same as any standard disk configuration. An IBM 3350, for example, has the following characteristics (GA26-1632-2, IBM Manual):

Average seek time
with movable heads - 25 ms

Average rotational
delay (latency) - 8.4 ms

Datarate - 1,198,000 bytes/sec

Storage Capacity - 317.5 megabytes

If a cartridge must be accessed the operations include (GA32-0038-1,IBM Manual):

1) Rewind the tape, update label, and unload the cartridge used in the previous request

2) Restore the cartridge to its cell, move the cartridge needed by the current job to the data recording device

3) Load the cartridge in the data recording device, verify the label and seek the data from the tape

4) Stage data to direct access storage device.

In order to simplify the staging process a uniform distribution with end points of 10 and 20 seconds will be used rather than trying to simulate the entire staging process. Once the data has been staged, it will be treated using IBM 3350 retrieval characteristics. Therefore when staging is required the retrieval time will be a value between 10 and 20 seconds inclusive plus retrieval time

from the IBM 3350. Only one cartridge retrieval at most will be required per job (see assumption 14).

In order to further simplify the I/O process the IBM 3350 retrieval time will be calculated as follows:

Disk retrieval time for a query (assumes clustering)
= average seek time + rotational delay (number of blocks)
+ block size (number of blocks)/
data transfer rate

Therefore if a query requires staging, the total MSS I/O time will be 10 to 20 seconds plus the results of the above equation. (If no staging is required then only disk I/O time is required.

In addition to the parametric models presented in this chapter for job interarrival rates, I/O and CPU requirements and MSS staging times, there are other factors that must be considered in a simulation. These factors include transmission rates, backend processor speeds, IFE processing time, and descriptor searches. These values are presented in Figure 19. Chapter IV gives an explanation of the architecture that is associated with these values.

Query Response Size

Because no historical data were available on the average size of a query response and because normally this value can have a very large range at any installation, a parametric model was developed which also has a wide range of values. Each backend processor will generate a response

* 1) Terminal to IFE Transmission Rate: 9600 baud

* 2) High Speed Bus Transmission Rate: 5 Mbytes/sec

**3) Vax Unibus Transmission Rate: 2 Mbytes/sec

**4) Broadcast Bus Transmission Rate: 1 Mbyte/sec

**5) Descriptor Processing Times: $\bar{x}$ = 30.896 ms σ = .0514 ms

**6) Processing and Formating Messages at Vax Controller and IFE: 8 ms

**7) Parse a Request Entering at Vax Controller and Formating a Response Leaving Vax Controller: 20 ms

**8) Address Generation: Triagonal Distribution -
Mode = 20 ms
High = 30 ms
Low = 10 ms

* Obtained from ECONET systems analysis
** Obtained from Hsiao's and Menon's MDBS Simulations (Menon,1981)

Figure 19 Additional Simulation Parameters

size taken from a triagonal distribution with the following parameters:

Low = 100.0 bytes

Mode = 5000.0 bytes

High = 20000.0 bytes

When all responses from the backends are assimilated the final response will be:

(number of backends X value from distribution) + 50 (header) bytes. This distribution is also a baseline model and will be changed to test its sensitivity.

Summary

This chapter has given an analysis of the current computer system at Eglin AFB and parametric models predicting the 1986 environment of the ECONET. The parametric models represent some of the input to be used in a simualtion of the ECONET. The next chapter will present a description of the backend database management system to be simulated as a part of the ECONET.

IV. BACKEND DATABASE SYSTEM DESCRIPTION

Introduction

The backend database computer system design chosen to be simulated in the ECONET environment is the Multi-backend Database System (MDBS) developed by Dr. David K. Hsiao and Dr. M. Jaishankar Menon at the Ohio State University (Menon, 1981). MDBS is a software implementation of a backend computer system. The design uses off-the-shelf equipment and therefore does not require specially built hardware (as opposed to a hardware backend processor). The design incorporates a number of slave minicomputers driven by duplicated database management software and is controlled by a master minicomputer. The goal of Hsiao's and Menon's research is to investigate whether the management of large databases using multiple minicomputers operating in parallel is feasible and desirable (Menon, 1981).

To date, MDBS has not been completely implemented, but simulation studies have shown it to have very fast response times. The researchers also claim that their design allows an almost unlimited addition of backend processors without performance degradation. Their simulations show that with the addition of each additional backend (to handle database growth) system throughput is increased. Previous research efforts by others showed an increase in throughput to a point followed by degradation with the addition of backend processors beyond that point.

This thesis incorporates an independent simulation of MDBS with modifications as required by the ECONET. Hsiao's and Menon's system, for example, does not use a mass storage system. The information gained from the system analysis and its extrapolation into 1986 parametric models (see Chapter 3) is used as input for the simulation. Many of the performance parameters such as backend processor speeds and transmission rates, however, are obtained directly from Hsiao's and Menon's model because the hardware being simulated is the same.

Justification for the Selection of MDBS

There are several reasons for choosing the MDBS design. First, MDBS uses off-the-shelf equipment. The backend processors are PDP 11's and the controller is a VAX 11. This equipment is also compatible with the VAX 11/780's used as IFE's in the ECONET. (Compatibility between equipment will be discussed in more detail later). Second, the MDBS is flexible in the sense that it allows for the easy addition of additional backend processors. The software at each backend procesor is identical and easily duplicated. Third, the potential of MDBS has been at least partially substantiated through detailed simulation studies at the Ohio State University. Fourth, a hardware backend database system is not currently available and is not predicted to be available until the late 1980's or early 1990's.

MDBS Systems Design

Figure 20 shows the basic components of MDBS and how it would fit in the ECONET. Hsiao and Menon set nine design goals for the MDBS (Menon, 1981). These goals include:

1) Avoiding channel limitation problems
2) Avoiding controller limitation problems
3) Backends must execute identical software
4) Communications among the backend processors and between a backend processor and a controller must be minimized
5) No special purpose hardware should be used
6) Support concurrent request execution
7) Overcome the device limitation problem by attaching more than one disk drive per backend
8) Design MDBS so that all backend's will participate in the execution of a request
9) Overcome the problem of data model limitation

Having established these design goals, Hsiao and Menon go through extensive efforts to prove that MDBS in fact does meet the goals. Rather than summarizing their efforts, proving how MDBS meets these goals, the reader is referred to Menon, 1981.

ECONET Upgrade Requirements versus MDBS Design Goals

Having chosen a basic design a question that now must be answered is, "Does the MDBS and the remainder of the ECONET meet the requirements for upgrade (see page 10)

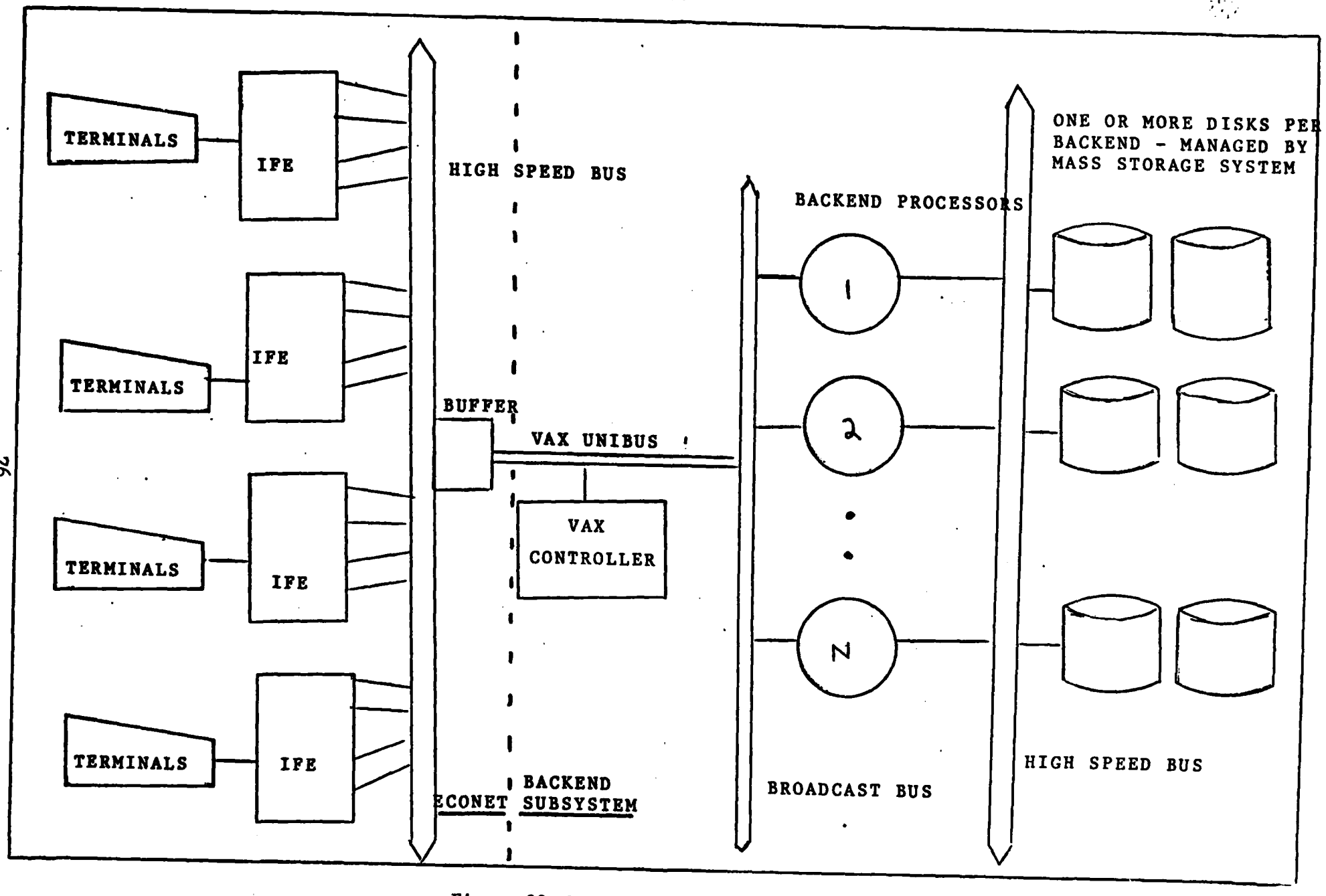


Figure 20 MDBS on the Eglin Computer Network.

established by Eglin personnel?"

Interactive - The IFE's will allow a greater number of terminals to be used on a computer system design specifically for interactive use.

Responsive - The addition of MDBS should increase productivity by giving more users faster turnaround and by also eliminating the need for many large printouts.

On-line - The MSS will eliminate operator intervention to mount tapes and search tape libraries.

Distributed - Queries should be run at the lowest possible level. Only transactions that are necessary should be executed.

Secure - MDBS will provide one path to the data. This should enhance the ability to validate database requests. MDBS also has its own authentication system.

Modular - As the database grows so can MDBS by adding additional backend processors.

Reliable - With IFE's and MDBS checking each other for erroneous messages, reliability should be increased. Also if one IFE goes down the others can pick up the slack.

It appears that MDBS's place in the ECONET does in fact help meet the requirements for upgrade. Having at least partially supported MDBS's place in the ECONET the data model used by the MDBS will now be discussed.

MDBS Data Model (Menon, 1981)

The data model chosen by Hsiao and Menon is the

attribute-based model. The other three data models (relational, hierarchical and network) are easily transformed into this model. All other transformations among the four models however can be cumbersome and/or inefficient. Hsiao and Menon wanted a data model for MDBS that overcame the problem of data model limitation. Any type of database structure can easily be represented using an attribute-based model.

The smallest unit of data in the attribute-based model is a *keyword* which is an attribute-value pair. The attribute represents the type, quality or characteristic of the value. A record is simply a collection of keywords. All the attributes in a record are required to be distinct. An example of a record might be:

(<file, STUDENT,>,<NAME,SELF>,<RANK,CAPT>,<AFSC,5135B>)

This record has four keywords with four unique attributes.

This type of data model is easily partitioned with no storage redundancy, as required by hierarchial and network data models. This is important because with MDBS the data in a file are partitioned as equally as possible across the backends and their associated disk units. This enables the parallel processing with the backends that is responsible for much of MDBS's predicted performance increase.

The attribute-based data model also has a relatively straight forward data manipulation language (DML) which will not be discussed here. The DML also allows concurrent

operations across several processors further enhancing parallel processing.

In addition to the attributed-based data model, record clustering techniques are used to narrow the search space and minimize the effort needed to retrieve records satisfying a given request. By organizing a database in clusters of data that are generally accessed together and by efficiently maintaining information about these clusters in a cluster definition table (CDT), response time can be further decreased.

As can be imagined the storage overhead for an attribute-based data model can be quite large. Every data value must have an attribute associated with it. The other three models also have overhead associated with them, but not nearly to the extent of the attribute-based model. With a mass storage device capable of handling over 470 billion bytes the overhead, however, is less significant. The flexibility that the attribute-based model provides in supporting various database structures and the easy implementation of parallel operations seem to offset its disadvantages.

Job Flow of a Database Transaction

This section describes the general database retrieval process of the ECONET with a MDBS system (A more detailed sequence of events is in Chapter V). Insertions, deletions, and updates will not be discussed here nor will they be simulated because of their similarity to retrievals

and also as part of an effort not to make the simulation unnecessarily complex.

1) A user enters a query at a terminal and transmits it via communications lines to an IFE.
2) The IFE performs any necessary preprocessing such as building a query package and message formatting and routes the query to the VAX controller via the high speed bus.
3) The controller parses the request and determines that it is a retrieval.
4) The controller broadcasts the request to all backends.
5) The backends will perform an equal amount of descriptor processing and address generation (identifying clusters) concurrently producing a list of secondary memory addresses of the tracks where the requested records are located.
6) The backends then broadcast their individual addresses obtained to every other backend. When all descriptor processing for a job has been done by all the backends then the query proceeds to the next step. While the idle processors are waiting for the other processors to finish, they can be handling other requests, thereby creating a multi-task environment.
7) The backends then generate sets of I/O operations and place them in the disk queues. The backend

processors talk to the Mass Storage System via the high speed bus on nodes separate from the VAX controller. As data are retrieved from the Mass Storage System it is further processed by the backend processors. This produces an environment where the backend processors are all concurrently reducing data while the disk units are also concurrently retrieving blocks of data. Each processor can be connected to several disk units with the data spread equally across each disk.

8) As each backend completes its portion of the processing it broadcasts the results to the VAX controller. The controller waits for all the backends to respond, properly formats the results and sends its answer to the IFE via the high speed bus.

9) The IFE determines which terminal initiated the request and sends the results to that terminal via communications lines.

Compatibility of the MDBS and ECONET

Eglin personnel have researched the interface requirements between the various components of the ECONET. Special software must be procured or written locally to interface the VAX IFE's with the high speed bus and to interface the high speed bus with the mass storage system. Similar interfacing must also be done between the high

speed bus and the backend system. The Mass Storage System should appear invisible to users. The backend processors will communicate directly with the Mass Storage System via the high speed bus on separate connections. This would allow for easier management and reduce the conflicts between scientific and database applications on the high speed bus. Because the exact complexity of the interface software is still undetermined by systems personnel at Eglin, delays due to software interface will be minimal in the simulation. Although much of the interface software promises to be complex, Eglin personnel are confident that it will not be impossible.

Summary

This chapter has presented a basic architectural description of MDBS and how it processes database requests. Also recognition is made of the fact that special interfacing will be required for various portions of the ECONET. The next chapter describes the simulation of the ECONET with a MDBS and gives an analysis of the results.

V. SYSTEM SIMULATION

Introduction

In order to help determine an optimal design for a backend database computer system for the ECONET, several simulations were run with various structural and parametric models. As mentioned in previous chapters, the parametric models were obtained from a systems analysis of the ECONET and from an investigation of Hsiao's and Menon's backend prototype (MDBS). The basic structural model is based on Hsiao's and Menon's MDBS with some necessary modifications to allow it to become a module in the ECONET.

The simulation language chosen to implement the simulation models was SLAM II. The simulation models used are a combination of two SLAM II modelling techniques: network modelling and discrete event modelling. The network portions of SLAM II allow one to simulate the flow of entities through a network as they encounter various timing delays and queuing situations. In the case of the backend database computer system simulated in this thesis, the flow of database transactions is simulated. The discrete event portion of SLAM II allows any number of tasks to be done at any point in time, thereby greatly expanding the capabilities of a basic SLAM II network-only model.

This chapter will first present the purposes of the simulation models to be followed by a detailed description

of the structural simulation model and how it represents a real world computer network. Next, the various measures of performance of the simulation models will be presented in order to provide a basis for evaluation of the various simulation results. The following section will then discuss the experimental design of the simulation runs. The experimental design is important because, if done properly, it allows the effects of changes to variables to be shown by the established measures of performance. In other words, the experimental design allows sensitivity analysis to be done. Finally, the simulation results will be presented, validated and analyzed to provide input to a final configuration for the backend database computer system for the ECONET.

Purpose of the Simulation Models

The primary purpose of the simulation models is to provide a realistic analysis of the behavior of a backend database computer system when added to the ECONET. A simulation should locate potential bottlenecks and provide accurate performance data. To obtain the data necessary to make design decisions, the following steps in model development and analysis were taken:

1) Development of a baseline model using systems analysis data and basic MDBS design concepts.
2) Proposal and observation of environmental perturbations to the baseline model (parametric changes).

3) Determination of major structural changes that are necessary to the baseline model and how these changes affect performance.

By following these steps, critical structural variables such as the optimal number of backend processors and/or disk units can be estimated more closely and provide a foundation for decisions about a final system configuration.

Structural Simulation Model

The various parametric models developed for simulation input were presented in Chapter III. This section describes the baseline structural simulation model and the steps a database transaction must go through before a user receives a response. Figure 21 shows a pictorial representation of the baseline structural model which consists of:

1) 4 IFE's each with 4 high speed channel paths
2) 1 VAX unibus
3) 1 VAX controller
4) 1 Broadcast bus
5) 2 backend processors each with 2 disk units

Each of the above items is considered a limited resource and modelled as such.

The network flow will be divided into several phases or modules to aid in the understanding of the flow of a database request. Each phase has one or more times that are collected for statistical analysis to help indicate

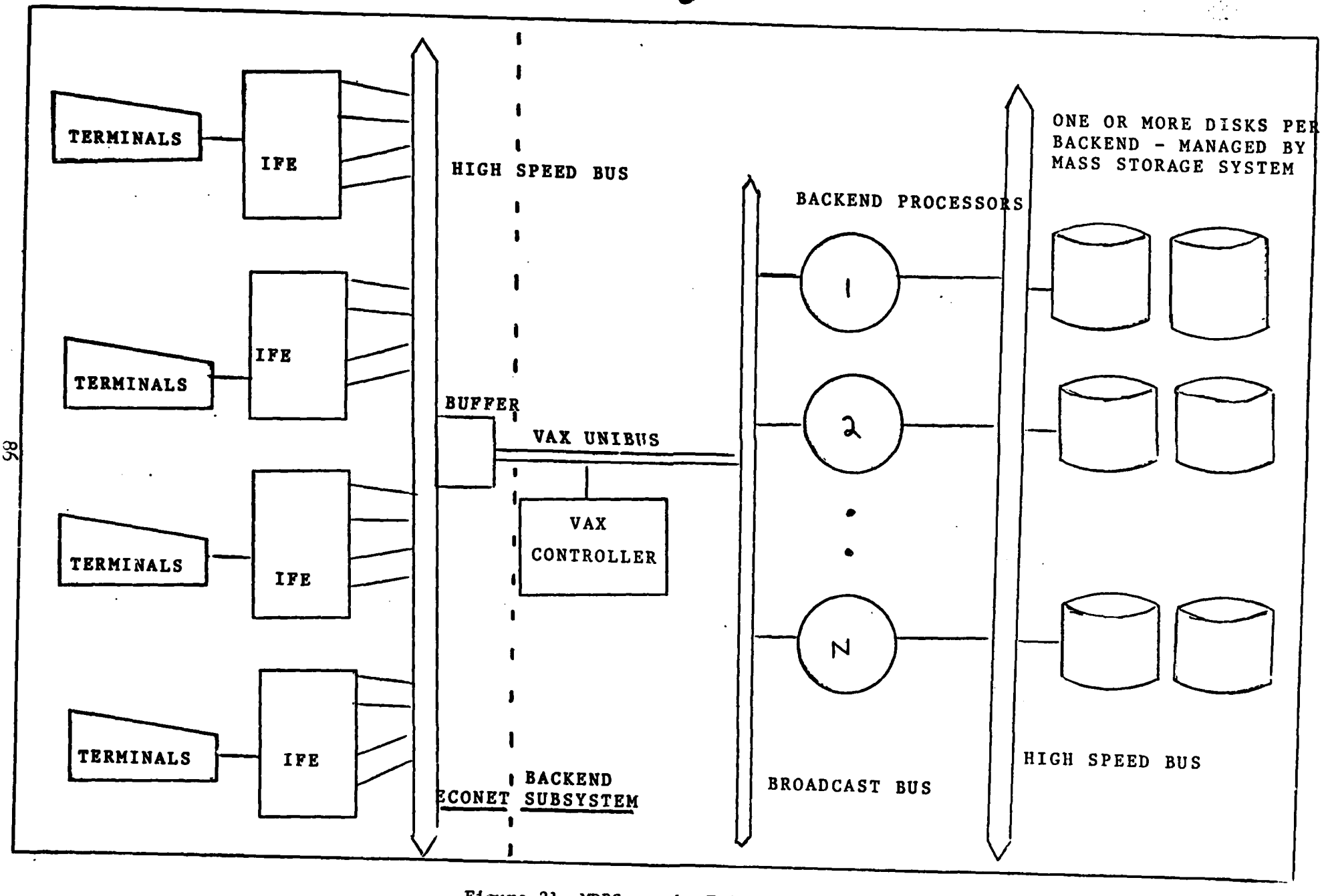
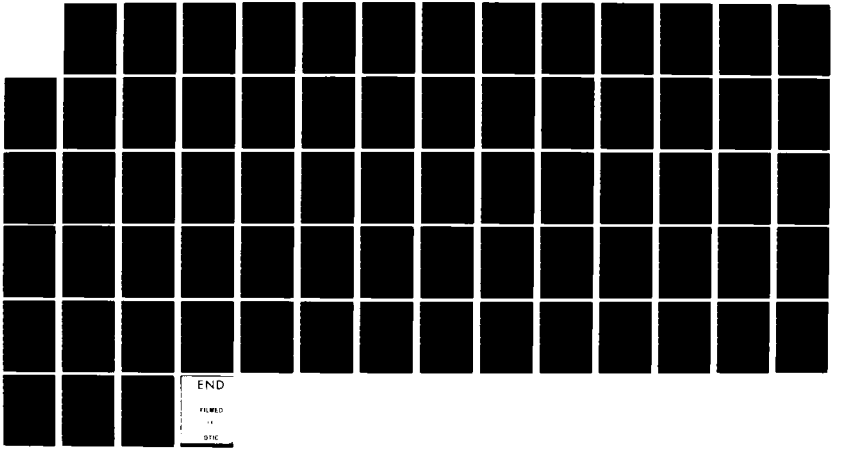
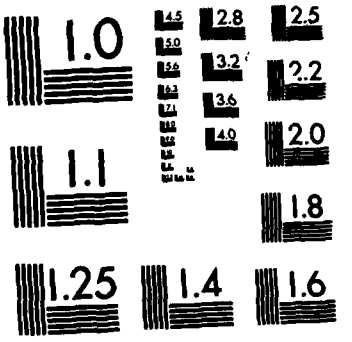


Figure 21 MDBS on the Eglin Computer Network

AD-A124 908 A BACKEND DATA BASE SYSTEM FOR THE EGLIN COMPUTER NETWORK(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING J L SELF NOV 82 2/2

UNCLASSIFIED AFIT/GCS/EE/82D-32 F/G 9/2 NL





possible bottlenecks.

Terminal and IFE Phase (Phase 1)

During this phase a user enters a query from a terminal. A subroutine determines what time of day it is (in milliseconds) and creates a job based on that time. During the hours of 0900 to 1700, for example, the job arrival rate is faster, but the input/output required and CPU required are less than other times of the day. These and other job parameters such as message size, response size, descriptor processing time, whether or not mass storage staging is required, and the block size for I/O are all determined as soon as a job enters the system. These attributes are carried with a job as it travels through the network and are used to produce the appropriate time delays. Chapter III explains how all parametric values used for simulation input were obtained.

Once a job enters the network, a delay is made for the transmission of the query from the terminal to the IFE buffer (memory) at 9600 baud. The message probabalistically selects one of the four IFE's, and seizes the IFE's CPU if it is available. Once the IFE is available an 8 ms delay is made for message formatting.

If a resource requested in the network is not immediately available, then the query (message) must wait until that resource is freed. The priority modelled for all resources is first in first out (FIFO). At any point in time several queries can be requesting the same resource

from various points in the network.

After processing at the IFE, the query must seize one of the four high speed bus paths emanating from each IFE. Once a line is available, the query will be transmitted over the high speed bus at 5 Mbytes/sec to a buffer at the node between the high speed bus and the VAX unibus. This buffering is necessary because of the difference in transmission rates of the high speed bus and the VAX Unibus (See Assumption 11, Chapter III). An estimated delay of 100 msec is made for the buffering

Parsing Phase (Phase 2)

The query has now arrived at the backend subsystem. After buffering between the high speed bus and VAX unibus the query is transmitted to the VAX controller's memory via the VAX Unibus. After obtaining the VAX controller's CPU the query is parsed. It is now ready to be broadcast to all the backends via the VAX Unibus and the broadcast bus. The query remains in the controller's memory until the VAX unibus and Broadcast bus are available.

Descriptor Processing Phase (Phase 3)

Once the parsed query exits the Vax controller it goes to the backend processors via the Vax unibus and the broadcast bus. A subroutine is called which places a message in each backend queue simultaneously to simulate a broadcast. Therefore the one message query has now become an N message query (N = number of backends). Each new

message must wait for its respective backend processor to become free in order to do descriptor processing. An equal share of the descriptor processing is done by each backend processor concurrently. Secondary memory addresses are generated by searching cluster definition tables (CDT). The CDT's allow mapping from query predicates to secondary memory addresses. When a backend is finished with its portion of the descriptor processing it obtains the broadcast bus and transmits the secondary addresses it has found to all the other backend processors.

When all descriptor processing for all backend processors has been transmitted to every backend, further processing of that query may continue. If a backend processor finishes its descriptor processing, it may handle requests from other queries. This creates a multiprogramming environment. The simulation model queues all descriptor processing responses until all responses for a particular query are received. A task (query) identifier is used to do this.

Input/Output Processing Phase (Phase 4)

Each backend processor resource now contains in its memory (queue) a set of secondary addresses for its attached disk units. The baseline model has two disk units (with IBM 3350 characteristics) connected to each backend processor via the high speed bus and managed by the mass storage system.

As described in Chapter IV, the data are equally

spread across the disk units. Each backend processor now concurrently generates a set of I/O operations and places them in the disk queues for concurrent retrieval. The simulation model simply takes the number of I/O blocks required that is associated with the query, divides it by the number of backend processors and places that number of I/O blocks in each disk resource queue. The simulation makes no attempt to model I/O channel contentions as each processor will have a separate channel to the mass storage system on the high speed bus. The last I/O operation generated for each backend is flagged to distinguish between I/O operations of different queries. A subroutine calculates disk retrieval times based on IBM 3350 performance parameters.

As data are retrieved from a disk, it are sent to the backend processor for further reduction (The CPU time required is equally distributed between I/O blocks). This creates a pipeline effect; blocks of data are being processed while other blocks are being retrieved. This sequence is also being repeated concurrently on the other backend/disk units. The retrieved and reduced data are buffered at the backend until all the generated I/O blocks associated with a backend have been retrieved and processed.

Response Phase (Phase 5)

When each backend processor has completed its share of the query processing its response is sent to the the VAX

controller via the broadcast bus and VAX unibus. Both buses are limited resources, so if they are in use the response must wait. Each backend's response is then held at the VAX controller until all backends have sent their portion of the query response. Again, the simulation uses queues to hold the responses until a match can be made from all backend on the task identifier.

After receiving all the individual responses, the VAX controller formats the complete response. The answer to the user's query is then routed via the VAX unibus and high speed bus to the IFE from which the query originated. Total response times are taken at the IFE (simulating a job buffered for print or tape output) and all the way back to the terminal (simulating a user viewing a response on a CRT).

Figure 22 shows all the resources available in the simulation model and summarizes all the activities competing for each resources from one query. This figure shows some of the potential bottlenecks in the network. The backend processors and VAX controller, for example, are candidates to be bottlenecks. The final simulation results should, however, indicate whether they actually do become bottlenecks.

Measures of Performance

In order to accurately measure the performance of the simulation model, several times are collected during the

	Resource (4) Interactive Frontends
Activities Competing	1) Message formatting into system
	2) Message formatting out of system
	Resource (1) VAX Unibus
Activities Competing	1) Message in for Parsing
	2) Messageout forDescriptor Processing
	3) Message in for responses for *each* backend
	4) Message out for final formatted response
	Resource (1) VAX Controller
Activities Competing	1) Parsing processing
	2) Assimulation and formatting of final response
	Resource (1) Broadcast Bus
Activities Competing	1) Message to backend for descriptor processing
	2) Message from *each* backend for descriptor processing results
	3) Message from *each* backend for query results
	Resource (2) Backend Processor
Activities Competing	1) Descriptor processing
	2) I/O operation generation
	3) Reduction of *each* block of data required per backend
	Resource (4) Disk
Activities Competing	1) I/O retrieval for each block of data required per disk

Figure 22
Baseline Resource and Activities that compete for them

various phases of a query. Also resource utilization statistics are monitored to help determine how busy a particular resource is during different periods of the day. Figure 23 gives a breakdown of the times measured during any simulation run and how they are related to the phases described in the previous section.

The resource utilization figures used for performance measurement describe the percent of time a given resource is busy during the period of measurement. If a simulation were run for the hours from 0900 to 1700, for example, a utilization rate of .65 would mean the resource was in use 65% of the time and idle 35% of the time. Average, maximum and minimum queueing delays and queue lengths for the various resources are also collected during a simulation run.

Experimental Design

Various simulation runs were made in an attempt to determine the structural and parametric models' sensitivity to change. Simulation runs will be made for one day periods, periods from 0900 - 1700, and from midnight to 0900. The following structural and parametric values will be candidates for sensitivity analysis:

1) Number of backend processors (structural)
2) Number of disk units (structural)
3) Job arrival rate (parametric)
4) I/O blocks required per job (parametric)
5) CPU time required per job (parametric)

Time	Description	Phase
1.	Transmission at 9600 baud from terminal to IFE	1
2.	IFE processing + transmission over high speed bus	1
3.	Buffering between high speed bus and VAX Unibus + VAX Unibus transmission time (in)	2
4.	VAX Controller parse time	2
5.	VAX Unibus transmission (out)	3
6.	Broadcast Bus transmission for descriptor processing (in)	3
7.	Descriptor processing and Broadcast transmission for all backends - includes waiting time for all backends to respond	3
8.	I/O processing - includes: MSS staging if required, address generation, all disk I/O, and processing of retrieved data	4
9.	Response assimilation - includes: Broadcast bus transmission of each backends response, waiting for all responses from all backends and formatting a final response	5
10.	Response time from Backend System to terminal includes: VAX Unibus transmission, high speed bus transmission, IFE formatting, and transmission back to terminal at 9600 baud	5
11.	Total system time all jobs	Total Time

Figure 23 Measured Times

Time	Description	Phase
12.	Total system time all jobs to IFE buffer (time 11 minus time to terminal)	Total Time
13.	Total system time for jobs requiring MSS staging	Total Time
14.	Total system time for jobs requiring MSS Staging - only to IFE buffer (i.e., time 13 minus time to terminal)	Total Time
15.	Total system time for jobs not requiring MSS Staging	Total Time
16.	Total system time for jobs not requiring MSS Staging - only to IFE buffer. (i.e., time 15 minus time to terminal)	Total Time

Note: All times will include waiting for unavailable resources

Figure 23 (cont'd)

Measured Times

6) Input message size (parametric)

7) Output response size (parametric)

The baseline parameters presented in Chapter Three provide a starting point in the simulations. The above values will be varied to see how their changes affect the simulation model's performance.

Structural Model Variation

Three structural models will be simulated. The first will be the baseline model already described. It has two backend processors each with two disk units, for a total of four. The other two structural models to be simulated will have three or four backend processors and six or eight disk units divided equally among the processors.

The basic purpose of these three models is to see if performance actually increases or decreases with the addition of backend processors and disk units and whether or not that increase or decrease is substantial enough to warrant the change in the structure. The MDBS design claims backend processors can be added to handle increased workloads without causing bottlenecks elsewhere, such as in communication paths. Also backend designs in general claim to provide slower degradation in response times the busier they get (up to a point) because of the greater degree of concurrent operations.

Parametric Model Variation

The parametric models represent the levels of various functions the simulation model must perform. By varying

these models, the real world environments of the ECONET such as increased workload, larger jobs and more complex queries can be simulated. This section describes the parametric variations to be made, why they will be made, and what the variations will hopefully discover.

Job Arrival Rate

By varying the job arrival rate we can evaluate a wide range of workloads. Any type of prediction of a workload five years in the future is shakey at best. By making a baseline estimate and then substantially changing the job arrival rate, we can hopefully come close to the actual future workload.

Job arrival changes also allow us to see how surges and prolonged increases in workload affect system performance. This is very important because we are interested not only in regular workloads, but we are also interested in at what level a system becomes saturated or unacceptable. We also want to know how capable the system is of handling temporary work increases.

Job arrival rates were varied as follows (times are in seconds):

		0900 to 1700 hrs	Other Hours
(Baseline)	1)	TRIAG (10,25,40)	TRIAG (200,445,690)
	2)	TRIAG (5,20,35)	TRIAG (100,222.5,345)
	3)	TRIAG (5,15,30)	TRIAG (5,111,172)

Note: TRIAG (Low, Mode, High)

The first parametric model was obtained as shown in

Chapter III. The other two models are simply increases in the baseline arrival rate. Further variations may be made if phenomenon of interest occur with these models.

Input/Output and CPU Time Required

I/O and CPU requirements of the system will actually be increased by increasing the job arrival rate. The I/O and CPU parametric models, however, will also be varied for each job to determine if jobs requiring larger amounts of I/O and CPU time have an adverse affect on response times. If they do, then a structural change may be necessary. With the data equally spread across each disk unit, as with the MDBS design, each disk unit and backend processor pair should carry an equal load. If there is a bottleneck with increased I/O and CPU requirements, then it should be a bottleneck spread over all backend/disk unit pairs. The addition of extra backend processors and/or disk units could be a possible solution in this case (assuming another bottleneck is not created).

The I/O and CPU models were varied during the 0900 to 1700 hours as follows:

	I/O Blocks	CPU Time (sec)
(Baseline)	TRIAG (15,25,35)	(12,36,60)
	TRIAG (20,30,40)	(18,42,66)
	TRIAG (25,35,45)	(24,48,72)

During the non 0900 to 1700 time frame, the extremely large I/O and CPU jobs will be run (see Assumptions Chapter III). Therefore the I/O and CPU models for non 0900 to

1700 jobs are varied as follows:

	I/O Blocks	CPU Time (sec)
(Baseline)	TRIAG (60,100,140)	TRIAG (24,72,120)
	TRIAG (120,200,280)	TRIAG (48,144,240)

With the changes in I/O and CPU proposed, the response times should necessarily increase. What the simulation hopefully will discover, however, is how much of that increase is due to just a larger I/O and CPU requirement and how much is due to queueing delays or bottlenecks. For example, if the CPU requirements for a job are increased from 30 seconds to 60 seconds we would expect to see at least a 15 second increase in response time for a two backend configuration plus, hopefully only a small amount of queueing delays.

Input Message Size and Output Response Size

Both of these values will be varied independently in an effort to discover if message traffic can become a bottleneck. Hsiao and Menon chose fast channels in order to avoid this problem. In the first backend prototype, for example, the communications paths proved to be a major bottleneck. The paths used in that prototype, however, were much slower than those modelled here. The following parametric variations will be made in input message size (in bytes):

(Baseline) TRIAG (150,350,550)

TRIAG (300,700,1100)

The second parametric model represents a doubling of

input message sizes. If message size could be a problem, any realistic input message should be within the parametric models being simulated. The following parametric variations will be made in the output response size (in bytes):

(Baseline) TRIAG (100,5000,20000)

TRIAG (200,10000,40000)

The second model here also represents a doubling in the message size. The values, if anything, represent the high range of a possible response message. Therefore if bottlenecks can occur in this range they should be quite evident.

The parametric variations just discussed will be made for each of the structural models described in the "structural model variation" section. Hopefully, the simulations will provide some insight into the ECONET's performance with a backend database computer system. The next section presents the results of the simulation runs and also gives an analysis of why the system (simulation) behaved as it did.

Simulation Results

The simulation results will first be presented separately for each structural configuration. The results for the baseline model will be discussed first, (two backends) followed by the results for the three and four backend structures.

Two Backend Model (Baseline Structural Model)-Results

All parametric models, where run with the two backend structural model. Variations in input and output message sizes were only made for the two backend structure. The reasons for this will be discussed later in this section.

Results From Variations of Job Arrival Rates

It is important that a backend database system be capable of handling various increases in workload. For this reason various workloads were run. Figure 24 presents turnaround times for various job arrival rates and the total number of jobs processed by the system. As can be seen from Figure 24 an increase in jobs during peak hours slightly increases turnaround times between rates 1 and 2. The difference between turnaround of rates 2 and 3, however, is quite dramatic. The system is clearly becoming saturated as the arrival rate increases beyond rate 2. The turnaround times during non-peak hours, however, remain the same as the job arrival rate increases. This is probably because during non peak hours the system is not yet saturated and the increase in arrivals have yet to reach the point where bottlenecks occur. This is in agreement with the general performance characteristics of other backend prototypes.

To further investigate why job turnaround has behaved as shown in Figure 24, the phases that compose total turnaround time for rate 1 will now be discussed. Figure

Arrival Rate 1 = Triag(10,25,40) seconds (Baseline)
Arrival Rate 2 = Triag(5,20,30) seconds
Arrival Rate 3 = Triag(5,15,30) seconds
Arrival Rate 4 = Triag(200,445,690) seconds
Arrival Rate 5 = Triag(100,222.5,345) seconds
Arrival Rate 6 = Triag(50,111,172) seconds

Times: Values are average(in milliseconds)

	Peak Hours (0900-1700)			Non-Peak Hours (0001-0900)		
	Rate 1	Rate 2	Rate 3	Rate 4	Rate 5	Rate 6
Time 11	34700	39800	90600	53300	52700	51800
Time 12	20800	25900	76000	39800	39200	38200
Time 13	49800	58200	121000	62900	62700	64400
Time 14	36700	44200	106000	50600	49900	50900
Time 15	32900	37800	87400	51100	50800	49900
Time 16	18900	23900	73400	37400	37300	36300
Total Jobs Processed	1159	1435	1729	70	143	289
Percent Increase in Jobs	-	23.8%	20.4%	-	104%	102%

Note: Triag is (Low,Mode,High) for a triagonal distribution.

Figure 24 Two Backends - Results - Turnaround Times for Various Job Arrival Rates

Note: Arrival Rates Same as Previous Figure (Figure 24).

Times: Values are average(in milliseconds)

	Peak Hours (0900-1700)			Non-Peak Hours (0001-0900)		
	Rate 1	Rate 2	Rate 3	Rate 4	Rate 5	Rate 6
Time 1	295	293	293	285	289	290
Time 2	8.07	8.07	8.07	8.07	8.07	8.07
Time 3	100.0	100.0	100.0	100.0	100.0	100.0
Time 4	20.0	20.1	20.0	19.6	19.8	19.8
Time 5	.180	.176	.178	.171	.173	.174
Time 6	.208	.201	.206	.196	.198	.199
Time 7	1840	5550	[illegible]00	16.0	16.0	16.0
Time 8	18400	19800	45500	39400	38700	37700
Time 9	39.6	39.5	44.5	39.0	39.1	39.3
Time 10	13900	14000	14000	13500	13500	13600

Figure 25 Two Backends - Results - Intermediate Times for Various Job Arrival Rates

25 presents the five phases discussed in the "Structural Simulation Model" section. The phases are also subdivided to further isolate the flow of a database request (See Figure 23). Each time will now be discussed for reasonableness (validation) and its effect on turnaround time. Also to determine if there were statistically significant differences between the following times, hypothesis testing was done. Figure 26 summarizes these tests. As can be seen, only one of the calculations were significantly different at a .05 level of confidence (i.e., the null hypothesis is never rejected).

Validation of Initial Results

Time 1 - This time is reasonable. The input message size is TRIAG (150,350,550) bytes. At a 9600 baud transmission rate the following minimum, modal and maximum values would be produced:

9600 baud 9600 bits/sec = 1200 bytes/sec

= 1.2 bytes/msec

$$\text{minimum value} = \frac{150 \text{ bytes}}{1.2 \text{ bytes/sec}} = 125 \text{ msec}$$

$$\text{modal value} = \frac{350 \text{ bytes}}{1.2 \text{ bytes/sec}} = 291.66 \text{ msec}$$

$$\text{maximum value} = \frac{550 \text{ bytes}}{1.2 \text{ bytes/sec}} = 458.33 \text{ msec}$$

H_o : Calculated value mean = Simulated value mean

H_A : Calculated value mean $\neq$ Simulated value mean

α = .05

$$Z = \frac{\text{Calculated value mean - Simulated value mean}}{\text{Standard Deviation of simulated value} \Big/ \sqrt{\text{Simulation Sample Size}}}$$

$$Z = Z_{.05/2} = Z_{.025} = 1.96$$

Simulation sample size = 1159 (rate 1)

Time	*Calculated Z	Reject H_o	Fail to Reject H_o
Time 1	1.6575	no	yes
Time 2	0	no	yes
Time 3	0	no	yes
Time 4	0	no	yes
Time 5	1.66882	no	yes
Time 6	1.411154	no	yes
Time 8	5.0505466	yes	no

* Means and standard deviations were from rate 1 simulations

Figure 26 Statistical Tests - Results

The values obtained from the simulation runs ranged from 285 to 295 milliseconds (rate 1) and are statistically close to the modal values calculated above.

Time 2 - This time is reasonable. The formatting at the IFE takes 8 millisecondss. The modal transmission time of the input message across the high speed bus would be:

Transmission rate = 40 megabits/sec

= 5 megabytes/sec = 5000 bytes/msec

$$\frac{350 \text{ bytes}}{5000 \text{ bytes/msec}} = .07 \text{ msec}$$

The value obtained from the simulation were all 8.07 milliseconds. The values for this entire time are such a small part of the overall turnaround time that they are insignificant. Without further investigation, however, the values do indicate there is no queueing associated with the IFE and the high speed bus.

Time 3 - This time is reasonable. The majority of this delay is due to buffering necessary between the high speed bus and the VAX Unibus. This time will never be less than 100 milliseconds because that was the buffering time established. The modal transmission time of the input message across the VAX Unibus would be:

Transmission rate = 2 megabytes/sec

= 2000 bytes/msec

$$\frac{350 \text{ bytes}}{2000 \text{ bytes/msec}} = .175 \text{ msec}$$

This is an extremely small value and an insignificant part of the total turnaround time. The simulation results obtained were 100.0 milliseconds. The .175 milliseconds were not included because of round off error in the SLAM II summary report.

Time 4 - This time is reasonable. The mean parse time for a query is 20 milliseconds with a standard deviation of 2 milliseconds. The average values obtained ranged from 19.6 to 20.1 milliseconds.

Time 5 - This time is reasonable. The values here should be the same as the transmission times of Time 3 (i.e., .175 milliseconds) as long as there are no queueing delays. The simulation results ranged from .171 to .180 milliseconds. Without further investigation, these values indicate there were no queueing delays associated with transmission out of the VAX controller on the VAX Unibus. Once again the delays associated with this time are small enough to be considered insignificant.

Time 6 - This time is reasonable. The modal transmission time for a message across the broadcast bus would be:

Transmission Rate = 1 megabytes/sec

= 1000 bytes/msec

$$\frac{350 \text{ bytes}}{2 \text{ Backends}} = 50 \text{ bytes (Header)} + 50 \text{ bytes (Header)}$$

= 200 bytes/message

$$\frac{200 \text{ bytes}}{1000 \text{ bytes/msec}} = .200 \text{ msec}$$

The message to each backend would include its share of the descriptor processing plus a 50 byte header. The simulation values obtained ranged from .196 to .208 milliseconds. These values indicate no waiting for the broadcast bus. The values are also small enough to be considered insignificant.

Time 7 - This time starts to show the saturation problem with the backend system. This time is the first indication of backend processor utilization.

Descriptor processing itself takes very little time (ie $\overleftarrow{x}$ = 30.896 = .0514 divided by the number of processors). The apparent delay during peak hours is caused by excessive waiting for the backend processors. The average wait times for both backend processors and other resources are shown in Figure 27 (arrival rates are the same as those in Figure 24 and 25). Comparing time 7 on Figure 25 with the wait times for the backend processors in Figure 27, one can easily see why descriptor processing takes so long. Another factor that contributes to the long delay during this time period is that the time includes the waiting delay for all backends to respond. Therefore, if

Note: Arrival Rates Same as Figure 24.

Times: Values are average(in milliseconds)

RESOURCE	Peak Hours (0900-1700)			Non-Peak Hours (0001-0900)		
	Rate 1	Rate 2	Rate 3	Rate 4	Rate 5	Rate 6
IFE 1	0.0	0.0	0.0	0.0	0.0	0.0
IFE 2	0.0	0.0	0.0	0.0	0.0	0.0
IFE 3	0.0	0.0	0.0	0.0	0.0	0.0
IFE 4	0.0	0.0	0.0	0.0	0.0	0.0
HS BUS 1	0.0	0.0	0.0	0.0	0.0	0.0
HS BUS 2	0.0	0.0	0.0	0.0	0.0	0.0
HS BUS 3	0.0	0.0	0.0	0.0	0.0	0.0
HS BUS 4	0.0	0.0	0.0	0.0	0.0	0.0
Vax Unibus	.001	.001	.001	0.0	0.0	0.0
Vax Controller	.008	.019	.016	0.0	0.0	0.0
Broadcast Bus	1.329	1.569	.669	.890	.913	.912
Backend 1	6646.9	7905.4	28113.8	16912.2	16827.1	16494.3
Backend 2	6648.7	7904.9	28111.1	16912.2	16827.1	16494.3
Disk 1	86.5	98.5	259.9	413.3	419.5	418.9
Disk 2	86.5	98.5	259.9	413.3	419.5	418.9
Disk 3	86.5	98.5	259.9	413.3	419.5	418.9
Disk 4	86.5	98.5	259.9	413.3	419.5	418.9

Figure 27 Two Backends - Results - Average Wait Time for Resources

one backend processor finishes in five seconds and another finishes in eight seconds then the time delay will be the greater of the two. The times, however, should be relatively close in value.

Examining the resource utilization statistics, it can be seen why the backend processors have such substantial wait times associated with them. Figure 28 depicts the average utilization of the resources that had any wait times. It now becomes clear that the bottleneck becomes the backend processor as arrival rates increase. Figure 26 indicates that the disk units also have substantial wait times. This is because all I/O operations are placed in the disk queues at once. This causes the last I/O operations to wait longer, thus driving up the average wait time for a disk. This really makes little difference, however, because the backend processors can not handle the retrieved data as fast as the disk units can retrieve it. This is shown in the low utilization rates of the disk units.

Time 8 - This time, like the previous time, shows the increased use of the backend processors. A typical job without any queueing delays would take the time shown in Appendix B. The first underlined value, however, does not account for the concurrent operations of the disk units with the backend processors. Since all the disk operations (except for the very first one) run concurrently

Note: Arrival Rates Same as Figure 24.

RESOURCE	Peak Hours (0900-1700)			Non-Peak Hours (0001-0900)		
	Rate 1	Rate 2	Rate 3	Rate 4	Rate 5	Rate 6
Vax Unibus	.0007	.0008	.001	.00001	.0001	.0001
Vax Controller	.0016	.002	.0024	.0001	.0002	.0004
Broadcast Bus	.0007	.009	.001	.00001	.0001	.0001
Backend 1	.66	.82	.99	.08	.16	.32
Backend 2	.66	.82	.99	.08	.16	.32
Disk 1	.0080	.0099	.0118	.0018	.0038	.0076
Disk 2	.0080	.0099	.0118	.0018	.0038	.0076
Disk 3	.0080	.0099	.0118	.0018	.0038	.0076
Disk 4	.0080	.0099	.0118	.0018	.0038	.0076

Note: Values represent the fraction of total time that a given resource is in use.

Figure 28 Two Backends - Results - Average Utilization of Resources

with the backend processors, the determinant of Time 8 is therefore the time it takes the two backend processors to handle the required CPU processing. The second underlined value in Appendix B is the true approximation for Time 8.

The CPU time slice before each I/O interrupt is simply the total CPU time required divided by the number of I/O blocks required. This means that in the simulation each CPU time slice will be the same for a job. The disks turn out to be much faster than the backend processor because of the large amount of CPU required. In other words, the Eglin jobs are CPU bound.

The rate 1 results for Time 8 in Figure 25 are an average of 18400 msec with a standard deviation of 7550 msec. Thus the calculated value in Appendix X is within .1483 standard deviations of the simulated value. Although Figure 26 shows that the calculated and simulated values are different statistically at the .05 level of confidence, the model for time 8 is believed to still be valid beause of round off error. The round off error inherent in the SLAM II simulation language can compound itself during time 8. This is the only period when multiple multiplications and divisions are occuring, thereby increasing the effect of round off error. For this reason the calculated value of time 8 shown in Appendix B is belived to be closer to the simulation value than is statistically shown.

All the wait times associated with the backend

processors in Figure 27 for rate 1 are simply CPU requests for the same job. The disks are so much faster than the backend processors that all backend processor requests for a job essentially arrive at the backend queue at the same time. With this type of queueing, the first request does not wait, the second request would wait one time slice, the third request would wait two time slices and so on. A typical job for rate 1 would, for example, wait the following average amount of time per backend:

$$\text{Time Slice} = \frac{36 \text{ seconds}}{25 \text{ blocks}} = 1.44 \text{ seconds}$$

One Backend's Share = 12.5 time slices

$$\text{Average Wait} \quad \frac{0 + 1(1.44) + 2(1.44) + \ldots 11(1.44)}{12}$$

= 6.781 msecs

The simulated value was 6646 msec.

This calculation shows that a job run at rate 1 is only waiting for itself. At rates 2 and 3 however, it can be seen that jobs are beginning to back up. Therefore at a rate faster than rate 1, jobs are beginning to wait on previous jobs. At rate 1, however, the majority of the jobs appear to completely finish before the next job arrives.

Since the backend processors are operating concurrently, they essentially split the required CPU time. A job that requires 36 seconds of CPU time, for example, would concurrently process 18 seconds on one backend and 18 seconds on the other backend for a total clock time of 18 seconds.

Time 9 - This time is reasonable. A typical job should take the following amount of time:

Total response message size = TRIAG (100, 500, 20000) bytes

Response size for a single backend = (50, 250, 10000) bytes

Formatting time is normal $\overleftarrow{X}$ = 20 ms = 2 ms

Broadcast bus transmission rate = 1 mbytes/sec

Wait time for all backend to respond (Z) - unknown

$$\text{Time 9} = \frac{\text{response size for a backend}}{\text{Broadcast Bus Rate}} + \text{wait for all backends to respond}$$

$$+ \text{ formatting of all responses}$$

$$= \frac{50 \text{ to } 10000 \text{ bytes}}{1 \text{ M bytes/sec}} + Z + 20 \text{ msec}$$

$$= .05 \text{ to } 10 \text{ msec} + Z + 20 \text{ msec}$$

The average simulated values obtained ranged from 39.1 to 44.5 milliseconds. This means the wait time for all backends to respond is approximately 10 to 20 milliseconds. These values seem reasonable because the backends have equal workloads and they receive their workloads at the same time. They should therefore finish processing a job very close to the same time.

Time 10 - This time is reasonable. There are little or no queueing delays associated with the transmissions from the backend system. The major portion of this time is accounted for by the transmission of the response from the IFE to the terminal at only 9600 baud.

The transmission over the VAX Unibus and high speed bus are an insignificant part of time 10.

From the results presented by the baseline model, with increases in job arrival rates only, the only problem seems to be the backend processors. It should be pointed out, however, that the baseline structural model can support baseline job arrivals (rate 1). Further investigations will now be presented for changes in I/O and CPU requirements and for changes in input and output message sizes.

Results From Variations in I/O and CPU Requirements

Although increases in job arrival rates will also increase the system's I/O and CPU requirments, separate simulations were made that increase the I/O and CPU requirements for each job. These increases will necessarily increase a job's turnaround time, but the question being investigated here is whether increases in I/O and CPU requirements create excessive and unanticipated delays. Job arrival rates are determined using baseline parametric models (rate 1).

Figure 29 shows all the times measured during the simulations. As can be seen the only substantial variations in times are those for I/O and CPU phases, descriptor processing and total turnaround time. This is to be expected since I/O and CPU requirements have been increased. No new bottlenecks are created at the controller, IFE's or any of the communications devices.

Arrival Rate = Triag(10,25,40) seconds (Baseline)

I/O Rate 1: I/O = Triag(15,25,35) blocks/job CPU = Triag(12,36,60) sec
I/O Rate 2: I/O = Triag(20,30,40) blocks/job CPU = Triag(18,42,66) sec
I/O Rate 3: I/O = Triag(25,35,45) blocks/job CPU = Triag(24,48,72) sec
I/O Rate 4: I/O = Triag(60,100,140) blocks/job CPU = Triag(24,72,120) sec
I/O Rate 5: I/O = Triag(120,200,280) blocks/job CPU = Triag(48,144,240) sec

Times: Values are average(in milliseconds)

	Peak Hours (0900-1700)			Non-Peak Hours (0001-0900)	
	*Rate 1	Rate 2	Rate 3	*Rate 4	Rate 5
Time 1	295	294	294	285	285
Time 2	8.07	8.07	8.07	8.07	8.07
Time 3	100.0	100.0	100.0	100.0	100.0
Time 4	20.0	20.0	20.0	19.6	19.6
Time 5	.180	.177	.177	.171	.171
Time 6	.208	.203	.202	.196	.196
Time 7	1840	3320	9770	16.0	16.0
Time 8	18400	21800	28300	39400	76900
Time 9	39.6	39.4	39.2	39.0	39.0
Time 10	13900	14100	13900	13500	13500
Time 11	34700	39700	52500	53300	90900
Time 12	20800	25600	38600	39800	77400
Time 13	49800	56100	69700	62900	99400
Time 14	36700	42000	55600	50600	87200
Time 15	32900	37800	50500	51100	88900
Time 16	18900	23700	36600	37400	75200

* Baseline model parameters

Figure 29 Two Backends - Results - All Times for a Query with Various I/O Rates

Figure 30, however, isolates the backend processor and disk resources and indicates that as I/O and CPU usage increases that once again the backend processors are becoming a slight bottleneck. Disk resources, however, even with heavy I/O requirments, are still relatively inactive.

Even though it is the backend processors that are becoming the bottleneck it is still important to notice that the processors can handle substantial increases in I/O and CPU requirements before noticable degradation during peak periods. Non-peak periods show no substantial degradation at all. A doubling of I/O and CPU requirements during non-peak hours less than doubles turnaround time. The more gradual increase during peak hours, however, does become noticable during rate 3 simulations. Variations in input and output message sizes will now be investigated.

Results From Variations in Input and Output Message Sizes

Because several other backend prototypes have experienced troubles with communications lines as bottlenecks, substantial increases in input and output message sizes were simulated.

Baseline input and output message sizes were doubled with extremely minor effects. Wait times for the various communications paths were either zero or in many cases so small that they were insignificant when compared to total turnaround times. Increases in job arrival rates and I/O and CPU requirements also put no noticable load on any

Rates same as Figure 29

Resource	Peak Hours (0900-1700)			Non-Peak Hours (0001-0900)	
	Rate 1*	Rate 2	Rate 3	Rate 4*	Rate 5
Backend 1	6646.94	8403.4	12712.1	16912.2	35292.6
Backend 2	6648.68	8405.1	12716.9	16912.2	35292.6
Disks	86.56	110.08	159.9	413.3	853.09

Average Wait Time

Resource	Peak Hours (0900-1700)			Non-Peak Hours (0001-0900)	
	Rate 1*	Rate 2	Rate 3	Rate 4*	Rate 5
Backend 1	.66	.77	.90	.08	.16
Backend 2	.66	.77	.90	.08	.16
Disks	.008	.01	.0112	.0018	.0027

Average Utilization of Resources

* Baseline arrival rates

Figure 30 Two Backends - Results - Backend and Disk Wait Times and Utilization with Various I/O Rates

communications lines (see Figures 25 and 29). For these reasons communication paths were not investigated further as possible bottlenecks.

Three and Four Backends Model Results

Having established some baseline results with a two backend processor architecture, three and four backend processor architectures were simulated. The results of the various three and four backend processor architecture simulations are presented in this section. The parametric models used in these simulations were the same as the baseline model. Only the structure was changed from two backends with two disks per backend to three and four backends with two disks per backend.

Results From Variations of Job Arrival Rates

Figure 30 presents the turnaround times for three and four backend processor structures. As might be expected, the turnaround times decrease as the number of backends increase. This can be seen by comparing the times in Figure 31 and by comparing Figure 31 with Figure 24. The reason for this apparent performance increase is obvious. There are more backend processors sharing the workload. Since with the MDBS design the processing is equally distributed across the backends and the disk units, then the addition of extra processors should increase response time. To determine exactly where in a database transaction this performance increase is occuring, the different phases

Job Arrival Rates Same as in Figure 24

Times: Values are average (in milliseconds)

	Peak Hours (0900-1700)			Non-Peak Hours (0001-0900)		
3 Backends	Rate 1	Rate 2	Rate 3	Rate 4	Rate 5	Rate 6
Time 11	26500	27600	28400	41200	41300	39900
Time 12	13000	13500	14400	27000	26500	25700
Time 13	41000	42500	45900	54100	54100	52200
Time 14	27300	28300	30800	40100	40200	39100
Time 15	25100	25900	26500	38500	39000	38100
Time 16	11400	11900	12600	24300	24000	23700
Total jobs	1150	1460	1755	75	153	299
% increase in jobs	-	26.9	20.2	-	104	95.4

	Peak Hours (0900-1700)			Non-Peak Hours (0001-0900)		
4 Backends	Rate 1	Rate 2	Rate 3	Rate 4	Rate 5	Rate 6
Time 11	23700	23900	24000	35100	35300	33900
Time 12	9680	9840	10000	20800	20400	19800
Time 13	38000	38600	38800	48800	48900	47900
Time 14	23800	24100	25000	34000	33900	33900
Time 15	22000	22200	22400	31800	32800	31900
Time 16	8060	8250	8370	17600	18000	17700
Total jobs	1143	1439	1734	77	151	296
% increase in jobs	-	25.8	20.5	-	96.1	96.0

Figure 31 Three/Four Backends - Results - Turnaround Times for Various Job Arrival Rates

of a database request can be examined in Figure 32. The performance increase seems to be coming during times 7 and 8.

A substantial decrease in descriptor processing (time 7) seems to show that database requests are not waiting as long for processor resources. Also I/O and CPU processing (time 8) is also considerably less. All other times, however, are very close to baseline runs, indicating no new bottlenecks. All performance changes seem related to the number of backend processors.

Figures 33 and 34 show that the number of backend processors is in fact the primary determinent of performance. Waiting times for processors have decreased substantially from the baseline model and utilization rates have all decreased. These figures substantiate the claim that with an "MDBS like" design, performance is increased with the addition of each backend processor. Although only up to four processors were simulated in this effort, it is clear that other resources such as communications paths and the controller still have plenty of capability remaining. This is shown by the fact that only the backend processor resources have a resources utilization rate higher than 1%.

Results From Variations in I/O and CPU Requirements

Having examined the performance of two, three, and four backend processor structural models for increases in job arrival rates, we know the potential bottleneck is the backend processors. For comparative purposes the three and

Job Arrival Rates Same as Figure 24

* Rates from baseline structural model (rate 1) for comparison

Times: Values are average (in milliseconds)

	Peak Hours (0900-1700)				Non-Peak Hours (0001-0900)			
3 Backends	Rate *	Rate 1	Rate 2	Rate 3	Rate *	Rate 4	Rate 5	Rate 6
Time 1	295	290	292	291	285	282	289	291
Time 2	8.07	8.07	8.07	8.07	8.07	8.07	8.07	8.07
Time 3	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Time 4	20.0	20.0	19.9	20.0	19.6	19.9	20.2	20.0
Time 5	.180	.174	.175	.175	.171	.169	.173	.175
Time 6	.208	.151	.155	.157	.196	.146	.149	.150
Time 7	1840	268	746	1440	16.0	10.9	10.9	10.9
Time 8	18400	12300	12300	12500	39400	26500	26000	25200
Time 9	39.6	32.8	32.9	32.8	39.0	32.7	33.6	33.3
Time 10	13900	14000	14100	14000	13500	14300	14800	14200

	Peak Hours (0900-1700)				Non-Peak Hours (0001-0900)			
4 Backends	Rate *	Rate 1	Rate 2	Rate 3	Rate *	Rate 4	Rate 5	Rate 6
Time 1	295	295	292	296	285	293	297	295
Time 2	8.07	8.07	8.07	8.07	8.07	8.07	8.07	8.07
Time 3	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Time 4	20.0	20.0	20.0	20.0	19.6	19.7	19.8	20.0
Time 5	.180	.177	.175	.177	.171	.176	.178	.177
Time 6	.280	.126	.125	.129	.196	.125	.127	.126
Time 7	1840	139	262	372	16.0	8.35	8.36	8.35
Time 8	18400	9170	9200	9270	39400	20400	20000	19400
Time 9	39.6	36.2	36.2	36.1	39.0	36.6	37.5	36.8
Time 10	13900	14000	14100	14000	13500	14300	14800	14200

Figure 32 Three/Four Backends - Results - Intermediate Times for Various Job Arrival Rates

Arrival Rates Same as Figure 24

* Rates from baseline structural model (rate 1) for comparison

Times: Values are average (in milliseconds)

3 Backends	Peak Hours (0900-1700)				Non-Peak Hours (0001-0900)			
Resource	Rate *	Rate 1	Rate 2	Rate 3	Rate *	Rate 4	Rate 5	Rate 6
Vax Unibus	.001	0	0	0	0	0	0	0
Vax Contr	.008	.009	.003	.021	0	0	0	0
Broadcast bus	1.329	.839	.903	1.010	.890	.866	.839	.784
Backends	6646	3642	3731	3911	16912	10602	10499	10417
Disks	86.56	48.26	48.55	49.9	413.3	267.3	268.4	266.4

4 Backends	Peak Hours (0900-1700)				Non-peak Hours (0001-0900)			
Resource	Rate *	Rate 1	Rate 2	Rate 3	Rate *	Rate 4	Rate 5	Rate 6
Vax Unibus	.001	0	0	0	0	0	0	0
Vax Contr	.008	0	.001	.003	0	0	0	0
Broadcast bus	1.329	1.454	1.563	1.635	.890	1.414	1.484	1.389
Backends	6646	2239	2271	2317	16912	7495	7624	7469
Disks	86.56	29.7	30.07	30.24	413.3	189.1	193.0	192.8

note: Wait times for all IFE's and High Speed Bus lines were zero and are not included in this figure.

Figure 33 Three/Four Backends - Results - Average Wait Times for Resources with Job Arrival Increases

Arrival Rates Same as Figure 24

* Rates from baseline structural model (rate 1) for comparison

Times: Values are average (in milliseconds)

3 Backends	Peak Hours (0900-1700)				Non-Peak Hours (0001-0900)			
Resource	Rate *	Rate 1	Rate 2	Rate 3	Rate *	Rate 4	Rate 5	Rate 6
Vax Unibus	.0007	.0005	.0007	.0008	.00001	.00001	.0001	.0001
Vax Contr	.0016	.0016	.002	.0024	.0001	.0001	.0002	.0004
Broadcast bus	.0007	.0005	.0007	.0008	.00001	.00001	.0001	.0001
Backends	.66	.4237	.5346	.6436	.08	.0551	.111	.215
Disks	.008	.0050	.0063	.0076	.0018	.0013	.0026	.0051

4 Backends	Peak Hours (0900-1700)				Non-Peak Hours (0001-0900)			
Resource	Rate *	Rate 1	Rate 2	Rate 3	Rate *	Rate 4	Rate 5	Rate 6
Vax Unibus	.0007	.0007	.0009	.001	.00001	.00001	.0001	.0002
Vax Contr	.0016	.0016	.002	.0024	.0001	.0001	.0002	.0004
Broadcast bus	.0007	.0007	.0009	.001	.00001	.00001	.0001	.0002
Backends	.66	.2984	.3775	.4561	.08	.0409	.0818	.1579
Disks	.008	.0035	.0045	.0054	.0018	.001	.0019	.0037

note: Average utilization rates for all IFE's and High Speed bus lines were zero and not included in this figure.

Figure 34 Three/Four Backends - Results - Average Utilization Rates for Resources with Job Arrival Increases

four backend structures will now be investigated for changes in I/O and CPU requirments as was the two backend structure. The focus here will not be whether or not there is a performance improvement, but rather "how much" of an improvement there is. As with the two backend model, the only significant variations in times were those measured during the descriptor processing phase and the I/O and CPU processing phase. For this reason the results presented will focus on these phases and the resources associated with them.

Figure 35 presents the various turnaround times for the indicated increases in I/O and CPU time. As noted in a previous section, the times will necessarily increase as the number of I/O blocks and CPU time increases. What is important to examine, however, is how much of the time increase is due to an increase in requirements and how much is due to queueing delays. Figure 36 shows that with the three or four backend structure, no significant new queueing delays are encountered with increases in I/O and CPU requirements. The biggest increase during peak hours is only slightly greater than one second (going from rate 2 to rate 3 - backend resource). A doubling of I/O and CPU requirements during non-peak hours only increases the average wait for a backend resource by less than four seconds. Figure 37 shows that the utilization of backend processors increases slightly, but as mentioned before, the increases in turnaround times and wait times for a backend

Job Arrival Rates are the Same as the Baseline Model Triag=(10,25,40)

* Baseline structural model values for comparison

I/O rate 1: I/O = Triag(15,25,35) CPU = Triag(12,36,60) Baseline
I/O rate 2: I/O = Triag(20,30,40) CPU = Triag(18,42,66)
I/O rate 3: I/O = Triag(25,35,45) CPU = Triag(24,48,72)
I/O rate 4: I/O = Triag(60,100,140) CPU = Triag(24,72,120) Baseline
I/O rate 5: I/O = Triag(120,200,280) CPU = Triag(48,144,240)

Times: Values are average (in milliseconds)

	Peak Hours (0900-1700)			Non-Peak Hours (0001-0900)	
3 Backends	*Rate 1	Rate 2	Rate 3	*Rate 4	Rate 5
Time 11	34700	29100	31900	53300	65600
Time 12	20800	15100	17800	39800	51500
Time 13	49800	43200	46000	62900	78900
Time 14	36700	29000	32200	50600	65000
Time 15	32900	27500	30200	51100	62800
Time 16	18900	13500	16200	37400	48700

	Peak Hours (0900-1700)			Non-Peak Hours (0001-0900)	
4 Backends	*Rate 1	Rate 2	Rate 3	*Rate 4	Rate 5
Time 11	34700	25500	27100	53300	52900
Time 12	20800	11300	13100	39800	38700
Time 13	49800	40900	41500	62900	66600
Time 14	36700	25800	26900	50600	51900
Time 15	32900	23800	25400	51100	49600
Time 16	18900	9600	11500	37400	35500

Figure 35 Three/Four Backends - Results - Turnaround Times with I/O and CPU Changes

I/O and CPU parameters same as Figure 35

Times: Values are average (in milliseconds)

* Baseline structural model values for comparison

3 Backends	Peak Hours (0900-1700)			Non-Peak Hours (0001-0900)	
Resource	*Rate 1	Rate 2	Rate 3	*Rate 4	Rate 5
Vax Unibus	.001	0	0	0	0
Controller	.008	0	.012	0	0
Broadcast bus	1.329	.868	.925	.890	.866
Backends	6646.9	4608.8	5696.8	16912.4	22541.9
Disks	86.56	61.9	76.15	413.3	559.84

4 Backends	Peak Hours (0900-1700)			Non-Peak Hours (0001-0900)	
Resource	*Rate 1	Rate 2	Rate 3	*Rate 4	Rate 5
Vax Unibus	.001	0	0	0	0
Controller	.008	.001	.006	0	0
Broadcast bus	1.329	1.505	1.55	.890	1.414
Backends	6646.9	2931.2	3756.9	16912.4	16242.2
Disks	86.56	39.9	50.5	413.3	403.07

Figure 36 Three/Four Backends - Results - Average Wait Times for Resources with I/O and CPU Changes

I/O and CPU parameters same as Figure 35

* Baseline structural model values for comparison

Times: Values are average (in milliseconds)

3 Backends	Peak Hours (0900-1700)			Non-Peak Hours (0001-0900)	
Resource	*Rate 1	Rate 2	Rate 3	*Rate 4	Rate 5
Vax Unibus	.0007	.0005	.0005	.00001	.00001
Controller	.0016	.0016	.0016	.0001	.0001
Broadcast bus	.0007	.0005	.0005	.00001	.00001
Backends	.66	.5046	.5916	.08	.112
Disks	.008	.0062	.0073	.0018	.0026

4 Backends	Peak Hours (0900-1700)			Non-Peak Hours (0001-0900)	
Resource	*Rate 1	Rate 2	Rate 3	*Rate 4	Rate 5
Vax Unibus	.0007	.0007	.0007	.00001	.00001
Controller	.0016	.0016	.0016	.001	.001
Broadcast bus	.0007	.0007	.0007	.00001	.00001
Backends	.66	.3588	.4369	.08	.0834
Disks	.008	.0044	.0053	.0018	.002

Figure 37 Three/Four Backends - Results - Average Utilization of Resources with I/O and CPU Changes

processor are not greatly affected. The three and four backend structural models seem to handle large I/O and CPU requirments adequately as well as increases in job arrival rates. The next section will make further comparison of all three structures and make recommendations for backend database computer system architecture for the ECONET.

Performance Comparisons of the Three Structural Designs

This section further evaluates and compares the three structural designs simulated. Because no specific performance criteria have been set for database systems on the ECONET, a selection of a specific design will be based on the relative ability of the three structures to handle the baseline workloads determined in Chapter III (job arrival rates and I/O and CPU requirements). The design selected should also not provide too much "overkill" by attempting to handle situations that are unlikely to occur. The focus will be on peak hour performance, as it has been shown to be more likely to tax the upper limits of the system's capability.

Turnaround Comparisons

Figure 38 gives a graphical comparison of turnaround for the three structures simulated. As expected, the more backend processors the faster the turnaround. What is important to note, however, is the rate at which turnaround increases as job arrival rate increases. The two backend model is clearly becoming saturated after a relatively small increase in the number of jobs processed. Admittedly

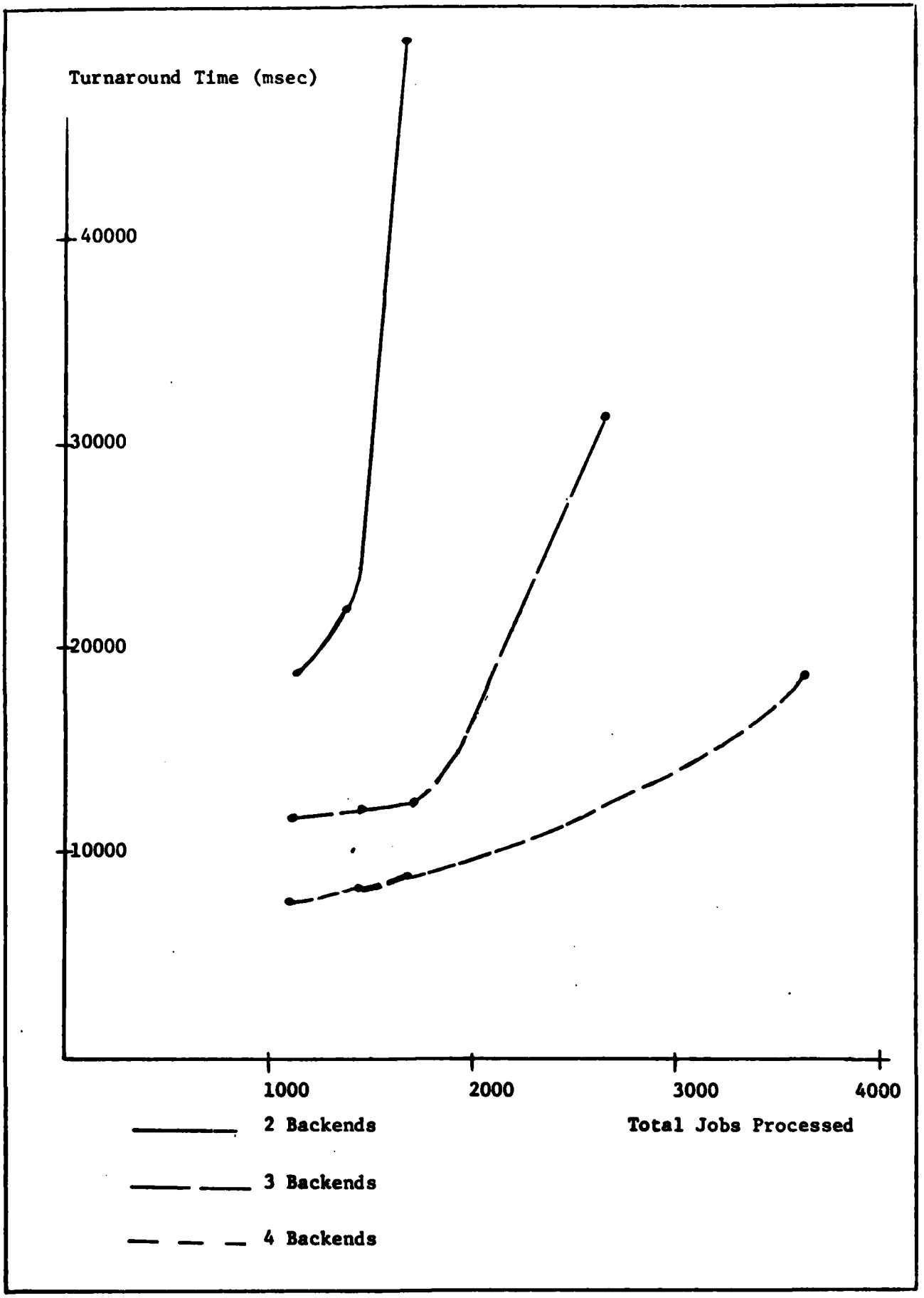


Figure 38 Peak Hours Turnaround Times for All Three Structural Models

the two backend structure can handle the baseline arrival rates, but it shows little growth potential. The three and four backend structures curves are much smoother and can handle substantial increases in jobs with little decrease in performance.

When Figure 38 is compared with Figure 39 no doubt remains as to which resource is the potential bottleneck. The backend processor wait time curve is the main determinent of turnaround time. This is made evident by the almost identical curves in Figures 38 and 39. Also in Figure 38, it can again be seen how the two backend structure is becoming saturated. Note, also, in Figures 37 and 38 that there is little difference between the three and four backend structures until the total jobs processed becomes a great deal larger. The systems performance capabilities are substantially increased with the addition of a third backend processor, but the addition of a four backend processor is not nearly as significant.

Figure 40 shows how busy the backend processors become with different job arrival rates. Ideally the processors should keep busy, but not so busy that excessive queueing delays are encountered. The two backend structure, for example, becomes too busy and creates delays. The three and four backend structures, however, become busy at a slower rate, therefore creating less of a delay than the two backend structure.

The two backend structures can handle baseline

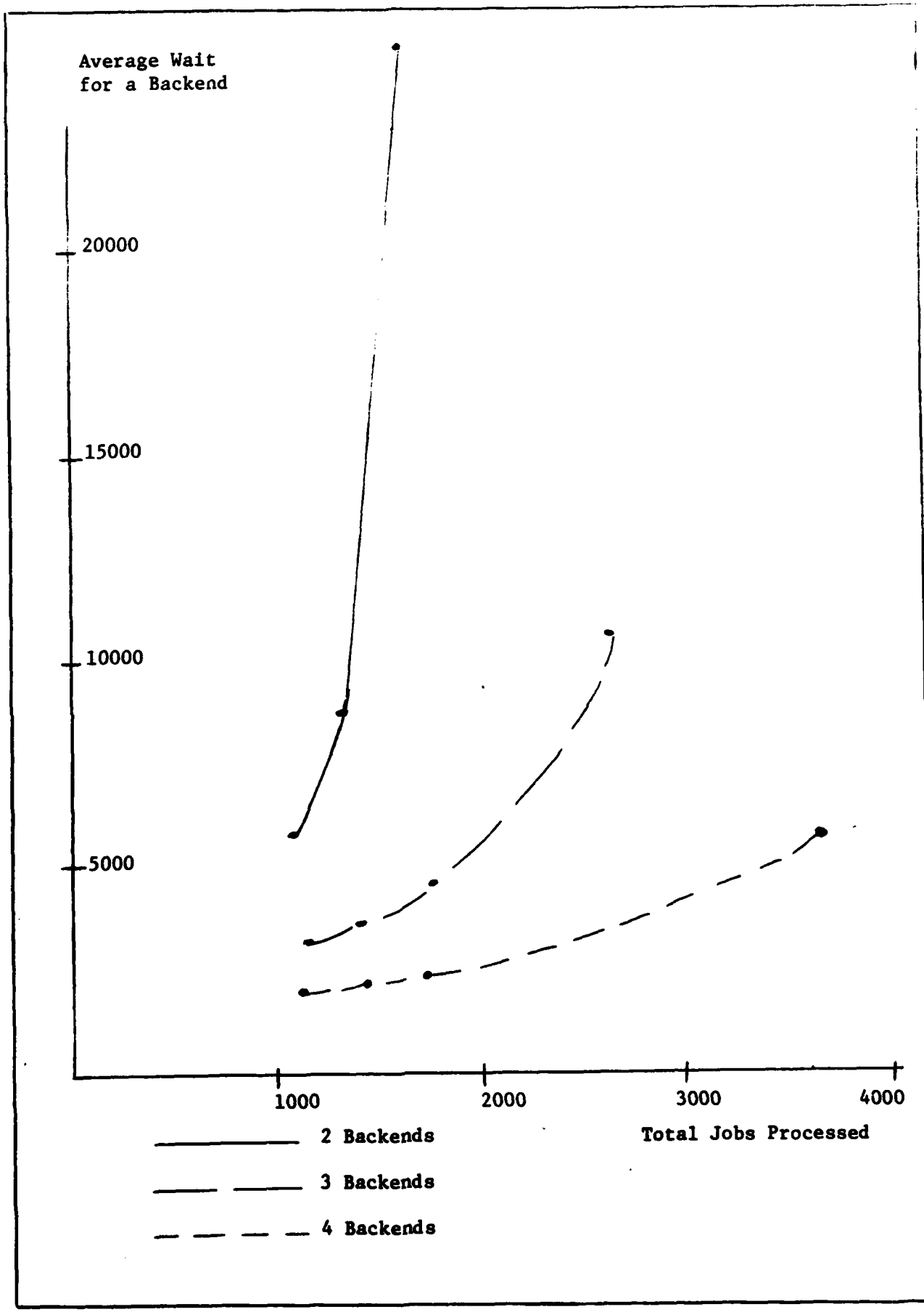


Figure 39 Peak Hours Wait Times for Backend Processors for All Three Structural Models

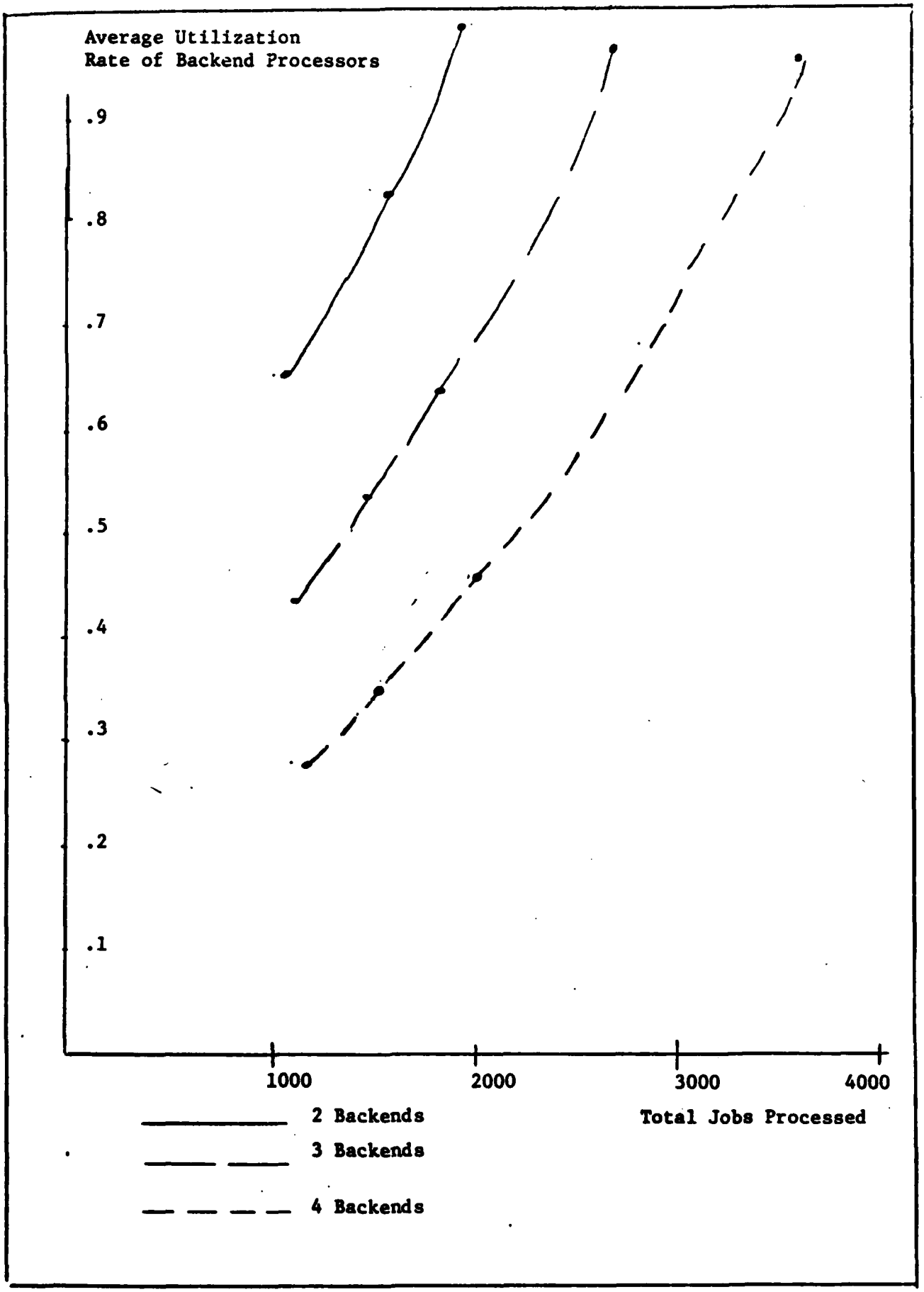


Figure 40 Peak Hours Utilization Rates for Backend Processors for All Three Structural Models

requirements, but just beyond a 23.8 per cent increase in jobs (i.e. beyond 1435 jobs processed), system's performance is rapidly degraded. The three and four backend structures, however, degrade at a much slower rate. The three backend structure can handle twice the jobs with only about a four second increase in average turnaround time. The four backend structure is even better, but does not provide as large of a marginal benefit as the change from two to three backends. Also it would probably be sometime before the outer capabilities of a four backend structure would be necessary. For these reasons the three backend structure is recommended for the ECONET.

Although a three backend structure has been recommended, because of MDBS design concepts, the addition of extra backend processors in the future would be easy. As requirements on the ECONET change, the relatively inexpensive backend processors could be procured. This would be considerably cheaper than other alternatives such as upgrading a mainframe computer on a system that did not use the backend concept.

Summary

This chapter has presented an indepth discussion of the simulation model of a backend computer system for the ECONET. A description of the flow of a data base transaction was given as well as the overall results of various scenarios that were simulated. It was found that performance does increase with the addition of backend

processors, but it was also found that the potential bottlenecks in an "MDBS like" design are the backend processors themselves.

A comparative analysis of two, three, and four backend processor structures determined that a three backend structure would best suit the database requirements for the ECONET in 1986. Additional backend processors can be added to the system beyond the data as requirements change.

VI. Conclusions and Recommendations

Overview

At the beginning of this investigation, four problems/goals were identified. The first problem was to analyze the computer system at Eglin AFB with a view towards incorporating a backend DBMS. The second goal was to include a backend DBMS design in the ECONET and model its performance using simulation. The third and final goal was to make recommendations on a final design of a backend DBMS for the ECONET.

The first problem was met by identifying computer workload characteristics and determining peak periods. The current configuration of the ECONET does not really show saturation in the area of data base applications. Discussions with persons in the Data Base section indicated that current turnaround was adequate. Analyzing future trends, however, showed a great potential for substantial increases in data base turnaround. The number of on-line data base transactions and terminal usage is expected to greatly increase future data base requests. It is possible, however, that the current system could have handled the projected increases in data base usage, but due to increased system usage in other applications, personnel at Eglin decided to improve their entire computer system. Since the Cyber would no longer be available for data base functions, a backend DBMS was recommended by Eglin personnel and was the object of this investigation.

The second goal was met by slightly modifying the Multi-backend Data Base System (MDBS) designed by David K. Hsiao and others, and including it in the ECONET. The design's performance was then simulated using SLAM II simulation techniques. The Backend design, however, was modelled independently of any other ECONET applications. This means the results found in this investigation could vary for values determined outside the backend subsystem. The values found for the backend subsystem itself, however, are believed to accurately reflect its performance characteristics. The simulations determined the bottlenecks in the backend system to be the backend processors themselves. The simulations also verified many of the performance characteristics of backend systems in general. Various changes in the structures simulated helped identify a configuration that will best benefit the ECONET in the future.

The third goal was met by a detailed analysis of the results from the various simulations. All three configurations simulated can adequately meet the baseline workload parameters. When the projected future workloads were simulated only two structures remained viable. The final decision was made based on relative performance of the two remaining structures and the anticipated future requirements of Eglin personnel.

Recommendations

The structural design recommended is one with three

backend processsrs and an MDBS-like backend system as shown in Figures 20 and 21. This configuration should efficiently meet the needs of ECONET database applications well into the future. Also with the MDBS design, additional processors can easily be added to improve the systems capabilities.

There are some other areas of interesting research on the MDBS and the ECONET that could be pursued:

(1) Simulation of MDBS in the ECONET with the other applications running concurrently. This would necessitate a deeper analysis of applications other than data base applications on the ECONET.

(2) Investigate how Hsiao's hardware implementations of a backend DBMS (DBC) could improve performance on the ECONET or any other computer system.

(3) Do a comparitive analysis of the MDBS/ECONET structure with Fonden's backend multiple-processor relational database computer system.

Final Comment

Backend data base management systems have received very much attention in the last eight years. They have shown a great potential in improving a computer system's ability to run many data base queries in a fast and efficient manner. An evolution, however, is already occuring in this relatively new area. Software versions of backend systems will probably soon be outdated. New hardware designs such as Hsiao's DBC show the capability to

increase performance several fold over even the best software backend designs. With these continuing changes that take advantages of the latest technological advances, the area of backend data base management systems should remain interesting for many years.

Bibliography

1. Baer, Jean-Loup. Computer Systems Architecture. Rockville, Maryland: Computer Science Press, Inc., 1980.

2. Banerjee, Jayanta and Hsiao, D. K. "Concepts and Capabilities of a Database Computer," ACM Transactions on Database Systems, 3(4): 347-384 (Dec 78).

3. Canaday, R. H. "A Backend Computer for Data Base Management," Communications of the ACM, 17(10): 575-583 (Oct 74).

4. Fisher, Paul S., Maryanski, Fred J. and Wallentine, Virgil E. "Evaluation of Conversion to a Backend Data-Base Management System," Proceedings of ACM 75 annual Conference, 293-297 (Oct 76).

5. Fonden, Robert W. Design and Implementation of a Backend Multiple-Processor Relational Data Base Computer System, Masters Thesis, Air Force Institute of Technology, Dayton, Ohio, 1981.

6. Heacox, H. C., Cosley, E. S. and Cohen, J. B. "An Experiment in Dedicated Data Management," Conference on Very Large Data Bases, 511-513 (Sep 75).

7. Hsiao, D. K. and Madnick, S.T. "Database Machine Architecture in the Context of Information Technology Evolution," Proceedings: Third International Conference on Very Large Data Bases, 63-84 (Oct 77).

8. Hsiao, D. K. "Data Base Computers," Advances in Computers, 19, 1-64 (June 81).

9. Hutchison, J. S. and Roman, W. G. "MADMAN Machine," Proceedings of the 4th Workshop on Computer Architecture for Non-numeric Processing, 85-90 (Aug 78).

10. GA32-0038-1. "IBM 3850 Mass Storage System Introduction and Preinstallation Planning," File no. 5370-07,1-23.

11. GA26-1638-2. "Reference Manual for IBM 3350 Direct Access Storage," File no. 5370-07, 1-17.

12. Kerr, Douglas S. "Database Machines with Large Content Addressable Blocks and Structural Information Processors, Computer, 12, 64-79 (March 79).

13. Lowenthal, Eugene I. "Backend Machines for Database Management: A Tutorial," Proceedings of the 5th Texas Conference on Computing Systems, 21-25 (Oct 76).

14. Madnick, Stuart T. "Trends in Computers and Computing: The Information Utility," Science, 19, 1197 (March 77).

15. Maryanski, Fred J. and Wallentine, Virgil E. "A Simulation of a Backend Data Base Management System," Proceedings of the 7th Pittsburgh Conference on Modelling and Simulation, 243-248 (April 76).

16. Maryanski, Fred J. "Backend Database Systems," Computing Surveys, 12, 3-23 (March 80).

17. Menon, M. Jaishankar and Hsiao, D. K. Design and Analysis of a Multi-Backend Database System for Performance Improvement, Functionality Expansion and Capacity Growth, Part I OSU CISRC-TR-81-7 (July 81), Part II OSU CISRC-TR-81-8 (Aug 81).

18. Misra, P. N. "Capacity and Analysis of the Mass Storage System," IBM SYST J, 20(3), 346-365 (1981).

19. Nagel, L. E. Systems Survey with a View Towards a Backend Database, Masters Thesis, Air Force Institute of Technology, Dayton, Ohio, 1978.

20. DAR-ATC-79-29. "High Speed Channel," Eglin DAR (FY79).

21. DAR-80-11. "Acquisition of a Large Mass Storage System for On-line Usage," Eglin DAR (FY80).

22. DAR-80-10. "Local Network fo Interactive Front End Computer Systems (Executive Summary)," Eglin DAR (Dec 79).

23. Directorate of Computer Sciences Planning Information. Eglin AFB, Florida, 1-59 (Sep 81).

APPENDIX A

Derivation of Projected Input Parameters (Baseline)

Values used in calculations:

1. Most recent job arrival rate = 16949 jobs in 6 months (10/1/81-3/31/82)
2. Percent increase per year in the number of jobs = 25% (4 years compounded multiple = 2.44 e.g. 1.25**4)
3. Multiplier for increase due to interactive use with more terminals which provide faster turnaround = 4
4. Percentage of 0900 to 1700 jobs = 90%

Peak Hour Job Arrival Rate =

$(16949) \times (1.25)^4 \times (4) \times (.90) = 148{,}965$ per 6 month period

Assuming in 6 months there are 128 work days then there are 1024 peak work hours or 3,686,400 peak work seconds, thereby giving a peak hour job arrival rate of $\frac{3{,}686{,}400}{148{,}965} = 24.7$ or approximately 25 seconds (1 job arrives every 25 seconds)

Because of the nonrigorous methodlogy used the variance from 25 seconds will be large. There a Triagonal distribution with the following parameters is used: Low = 10 sec, Mode = 25 sec, high = 40 sec.

Non-Peak Hour Arrival Rate =

$(16949) \times (1.25)^4 \times (4) \times (.10) = 16551$ jobs per 6 month period

Assuming 128 work days there are 2028 non-peak work hours. Calculating as above a job arrives every 445 seconds. Therefore a triagonal distribution of Low = 200 sec, Mode = 445 sec, High = 690 sec will be used.

Peak Hour CPU Usage Required =

Values used in this calculation:

1. Average CPU usage per job (over past 5 years) = $\frac{\frac{\text{CPU Hours} \times 60 \text{ minutes}}{\text{number of jobs}} + \ldots}{N}$

$$= \frac{\left(\frac{159.36 \times 60}{16949}\right) + \left(\frac{116.20 \times 60}{18718}\right) + \left(\frac{211.19 \times 60}{15508}\right) + \left(\frac{214.11 \times 60}{13875}\right) + \left(\frac{81.31 \times 60}{5321}\right)}{5}$$

= .71928 minutes

2. Percent reduction due to assumption 11 (Chapter 3) = 85%

Peak hour usage per job = .71928 min approximately 36 seconds.

Using a triagonal distribution with a wide variance we have:

Low = 12 sec, Mode = 36 sec, High = 60 sec.

Non-Peak hour CPU Usage Required =

By assumption 8 (Chapter 3) I/O bound jobs will be run during non-peak hours. Increased I/O will also result in increased CPU time to process the larger volumes of data. Therefore all parameters for peak hours will be doulbled for non-peak hours giving: Low = 24, Mode = 72, High = 120 seconds.

Peak hour I/O Required =

Using the same equation as the CPU usage required for peak hours with I/O hours replacing CPU hours we have:

Average I/O usage per job (over past 5 years) =

$$\frac{\frac{816.63 \times 60}{16949} + \frac{631.24 \times 60}{18718} + \frac{283.90 \times 60}{15508} + \frac{168.97 \times 60}{13875} + \frac{62.73 \times 60}{5321}}{5}$$

= 1.49 minutes

Assuming the percent reduction for assumption 10 (Chapter 3) is 50% and the percent reduction because of the mass storage system eliminating waiting for operators to hang a tape is 50% then:

(1,49 min x 60)(.5)(.5) = 22.35 I/O seconds per job

Assuming one block of I/O on the average is 1000 bytes (ie. 100 byte records blocked 10 to 1) then one block of I/O takes:

Average seek time + average rotational delay + $\frac{\text{block size}}{\text{data rate}}$

= 25 msec + 8.4 msec + $\frac{1000}{1198}$ = 868 msec per block of I/O

Continuing: $\frac{22.35 \text{ sec} \times 1000 \text{ sec/msec}}{868 \text{ msec per block}}$ = (approximately) 25 blocks of I/O per jol

As with the other input parameters calculated, the distribution will be assumed to have a wide variance. A triagonal distribution with the following parameters will be used: Low = 15, Mode = 25, High = 35 blocks.

Non-Peak hour I/O usage required =

Because of assumption 8 (Chapter 3) only I/O intensive jobs will be run at non-peak hour. Therefore all parameters for the peak hour parametric model will be multiplied by a factor of 4 to establish a baseline model for non-peak hour I/O usage giving: a triagonal distribution with Low = 60, Mode = 100, High = 140 blocks of data.

APPENDIX B

Time 8 Calculations for a Typical Job:

Values used in calculations:

1. Modal CPU time required = 36 seconds
2. Modal Disk blocks required = 25 blocks
3. Average disk block size = 1000 bytes
4. Mass storage staging required on 10% of retrievals. Staging time comes from a uniform distribution with endpoints of 10 and 20 seconds. An average of 15 seconds staging time will be used.
5. Average disk seek time = 25 msec
6. Average rotational delay =8.4 msec
7. Data transfer rate for disk = 1,198,000 bytes/sec
8. Number of backends = 2
9. Modal address generation = 20 msec

$$\text{Time 8} = \frac{\text{Blocks of data}}{\text{Number of backends}} \times \left(\text{Average seek time} + \text{Average Rotational delay} + \frac{\text{Average Block si}}{\text{Data tra rate}} \right.$$

$$+ \frac{\text{CPU time required}}{\text{Number of backends}} + (.1 \times 15 \text{ seconds}) + \text{address generation}$$

$$= \frac{25 \text{ blocks}}{2 \text{ backends}} \times \left(25 \text{ msec} + 8.4 \text{ msec} + \frac{1000 \text{ bytes}}{1{,}198 \text{ bytes/msec}} \right)$$

$$+ \frac{36 \text{ seconds}}{2} + 1.5 \text{ seconds} + 20 \text{ msec}$$

= 19947.93 msec per typical job (non concurrent disk and backend operation

With concurrent disk and backend operations:

$$19947.97 - 25 - \frac{1}{2} \quad 25 + 8.4 + \frac{1000}{1198} = 19520.036 \text{ msec per typical job}$$

Appendix C

Baseline Slam II Network Diagram:

This appendix presents a Slam II network diagram of the baseline structural model. The symbols used are described in Introduction to Simulation and SLAM by A. Alan B. Pritsker and Claude Dennis Pedgen. A list of attributes and resources are also given in this appendix as well as in the source listing. Several events and user functions are also called in the network. These events and user functions are availiable in documented source code format.

Attributes:

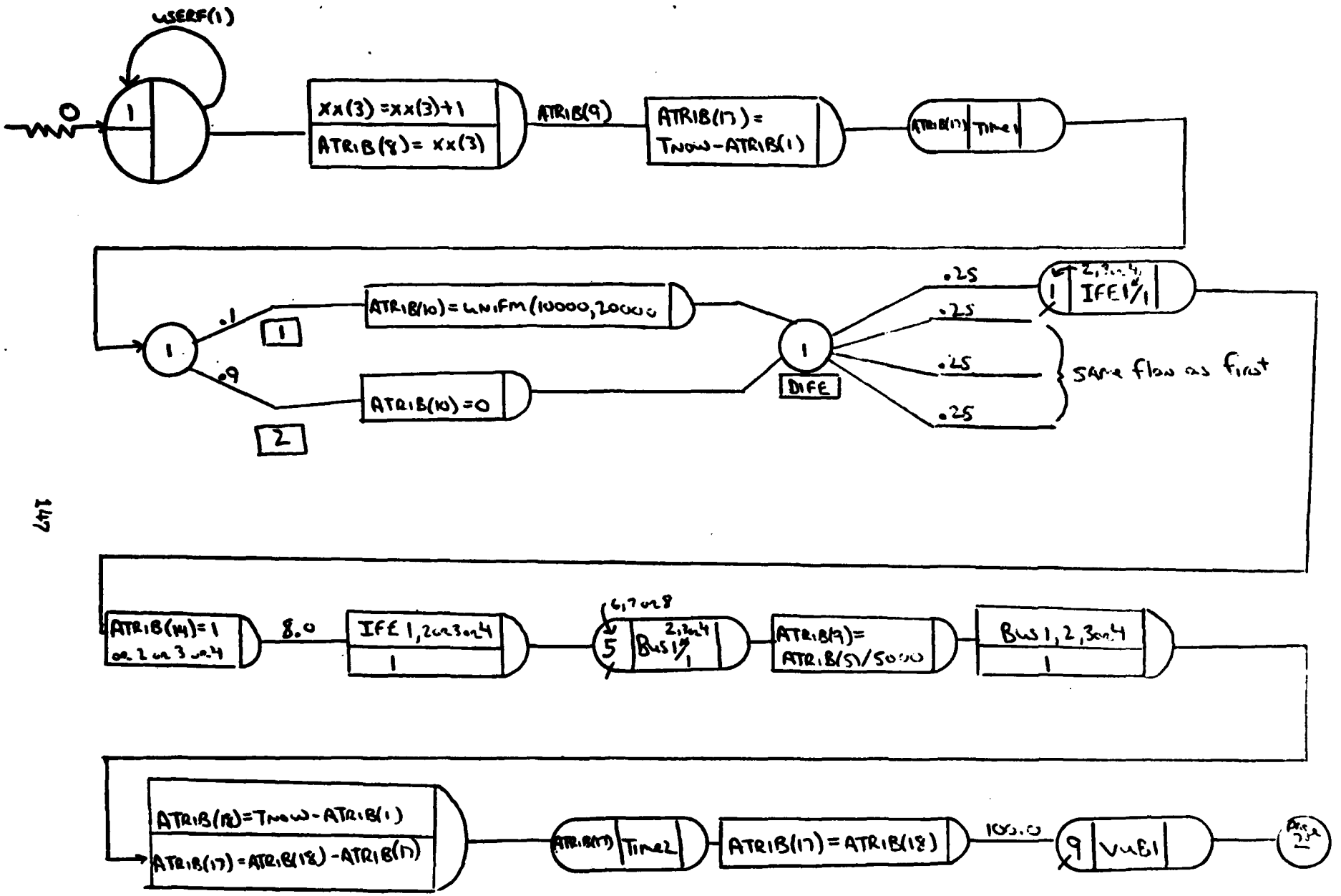
1. Arrival Time
2. CPU time required
3. I/O blocks required
4. Message type
5. Message size
6. Descriptor processing time
7. Backend processor being used
8. Task number
9. Time delay for various activities
10. Mass Storage system indicator for staging
11. Descriptor processing time for a backend
12. I/O blocks required per disk unit
13. Indicator for last block of I/O
14. IFE used
15. Block size for I/O blocks (bytes)
16. Response message size for a backend and total response size for later in processing

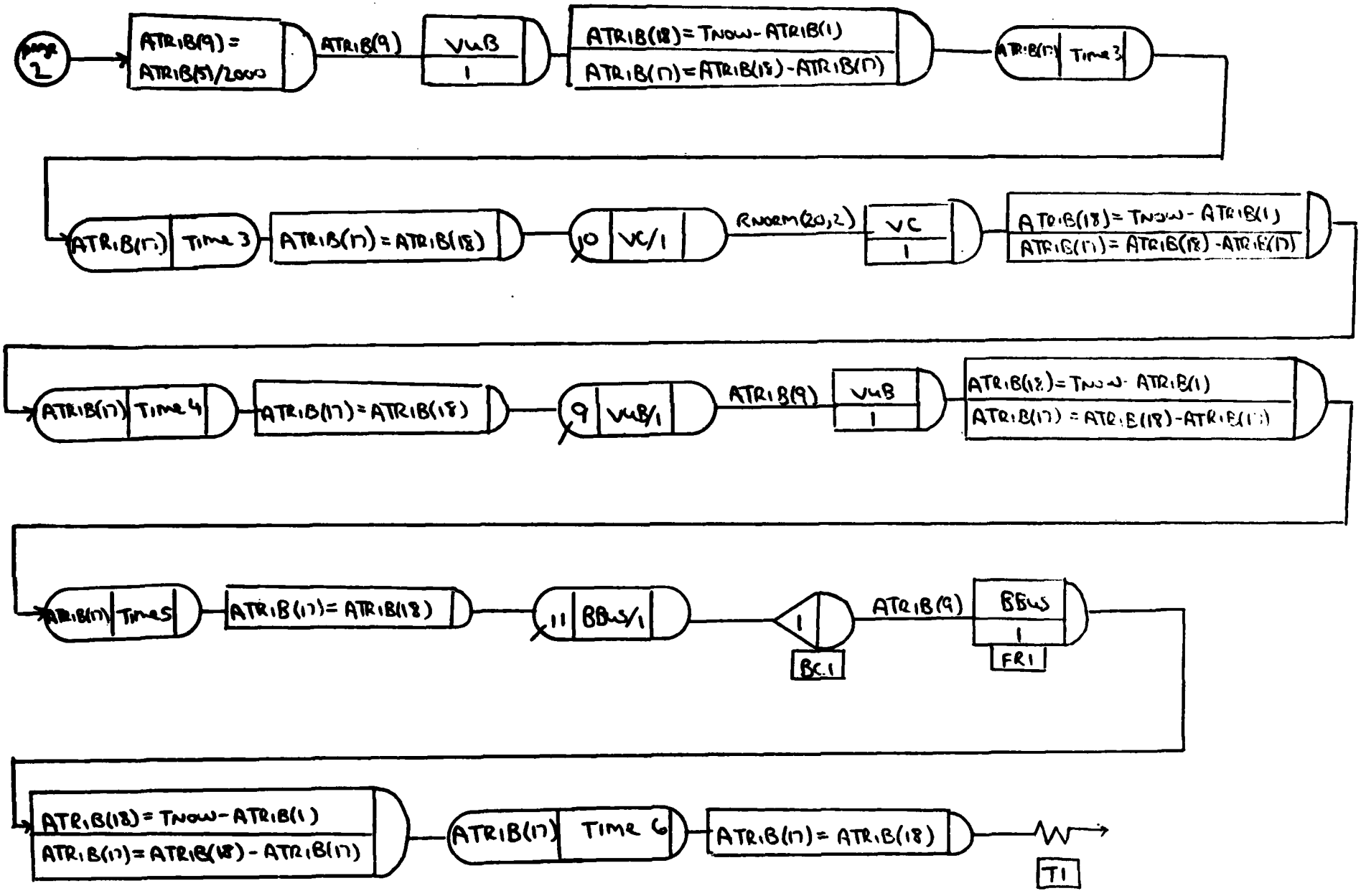
17-19. Used for collection of various times

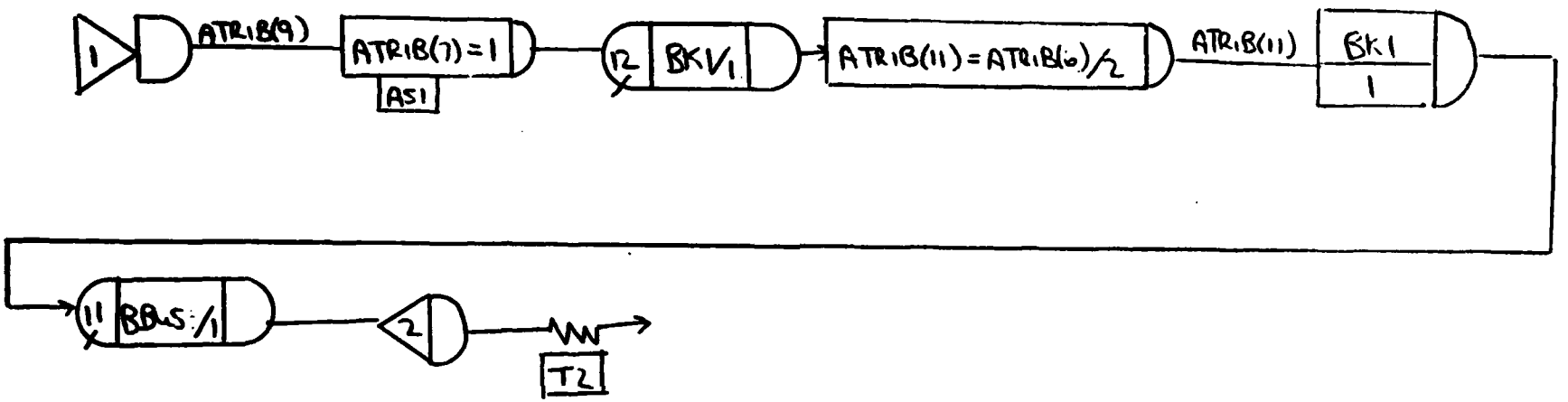
Resources:

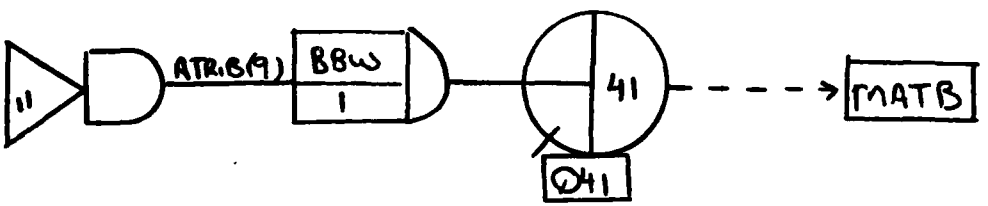
	Label in Network
IFE's (4)	IFE1-4
High Speed Bus (16)	Bus1-4 (4 each)
Vax Unibus (1)	VUB
Vax Controller (1)	VC
Backends (2)	Bk1 and Bk2
Disks (4)	Disk1-4

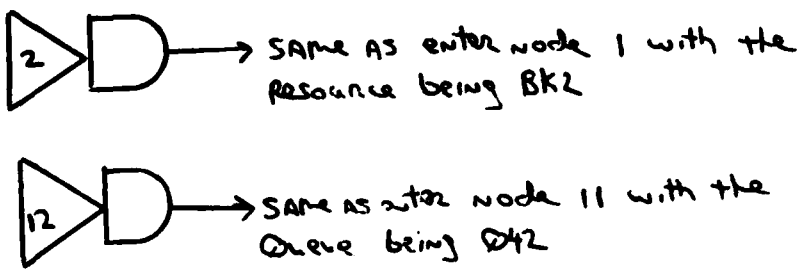
Note:
Some parametric values are assigned in the events and user functions and some parametric values are assigned in the network. Both must be examined to understand how the simulation functions.

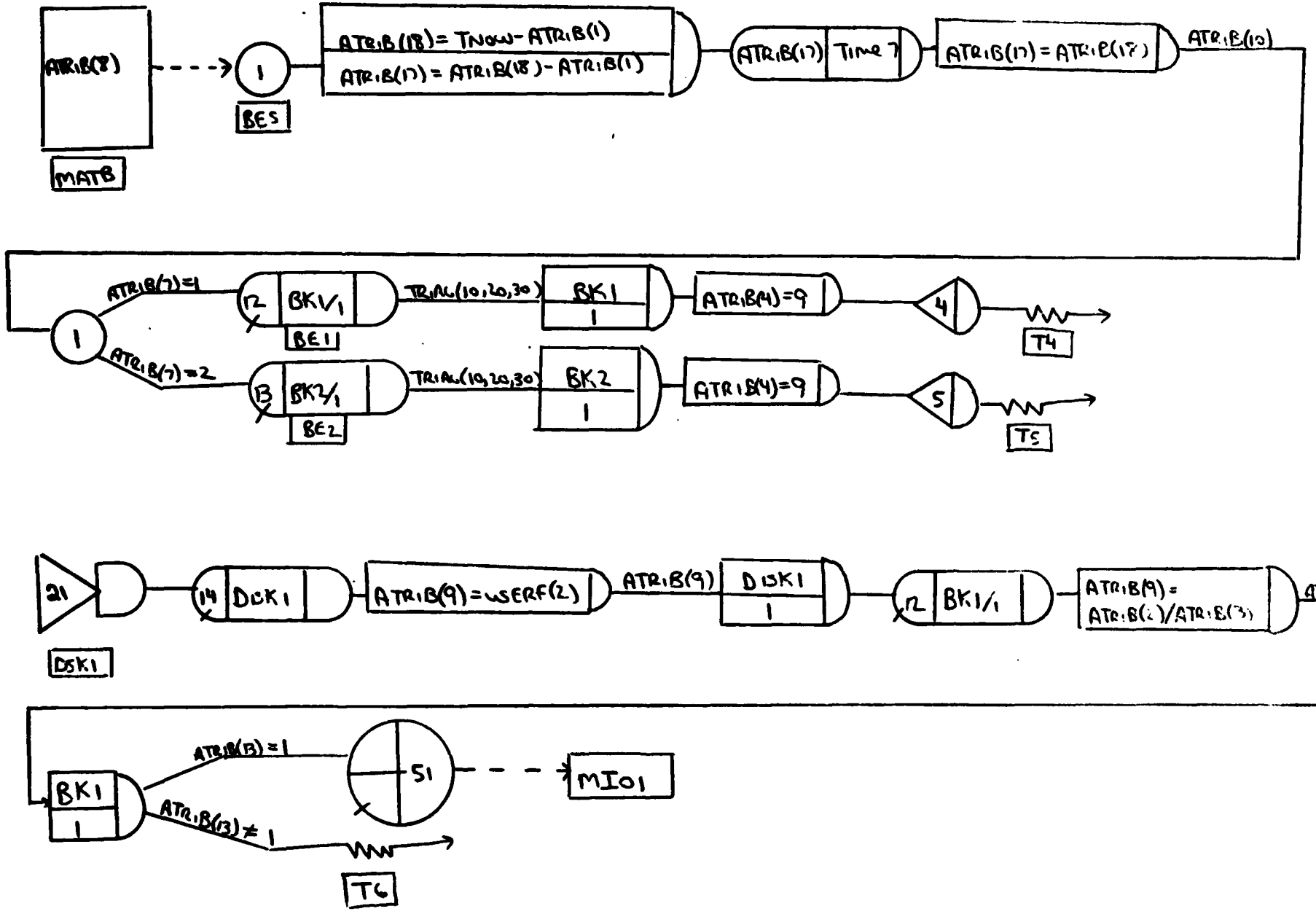

USERF(1)
0
1
XX(3) = XX(3)+1
ATRIB(8) = XX(3)
ATRIB(9)
ATRIB(17) = TNOW - ATRIB(1)
ATRIB(17) TIME 1
1
.1
1
ATRIB(10) = UNIFM(10000,20000)
.9
2
ATRIB(10) = 0
1
DIFE
.25
.25
.25
.25
1
IFE 1/1
same flow as first
ATRIB(14) = 1 or 2 or 3 or 4
8.0
IFE 1,2 or 3 or 4
1
6,7 or 8
5
BUS 1/1
ATRIB(9) = ATRIB(5)/5000
BUS 1,2,3 or 4
1
ATRIB(18) = TNOW - ATRIB(1)
ATRIB(17) = ATRIB(18) - ATRIB(17)
ATRIB(17) TIME 2
ATRIB(17) = ATRIB(18)
100.0
9
VUB1


page 2
ATRIB(9) = ATRIB(15)/2000
ATRIB(9)
VuB
1
ATRIB(18) = TNOW - ATRIB(1)
ATRIB(11) = ATRIB(18) - ATRIB(11)
ATRIB(11) Time 3
ATRIB(11) Time 3
ATRIB(11) = ATRIB(18)
10 VC/1
RNORM(20,2)
VC
1
ATRIB(18) = TNOW - ATRIB(1)
ATRIB(11) = ATRIB(18) - ATRIB(11)
ATRIB(17) Time 4
ATRIB(17) = ATRIB(18)
9 VuB/1
ATRIB(9)
VuB
1
ATRIB(18) = TNOW - ATRIB(1)
ATRIB(17) = ATRIB(18) - ATRIB(17)
ATRIB(17) Time 5
ATRIB(17) = ATRIB(18)
11 BBuS/1
1
BC.1
ATRIB(9)
BBuS
1
FR1
ATRIB(18) = TNOW - ATRIB(1)
ATRIB(17) = ATRIB(18) - ATRIB(17)
ATRIB(17) Time 6
ATRIB(17) = ATRIB(18)
T1

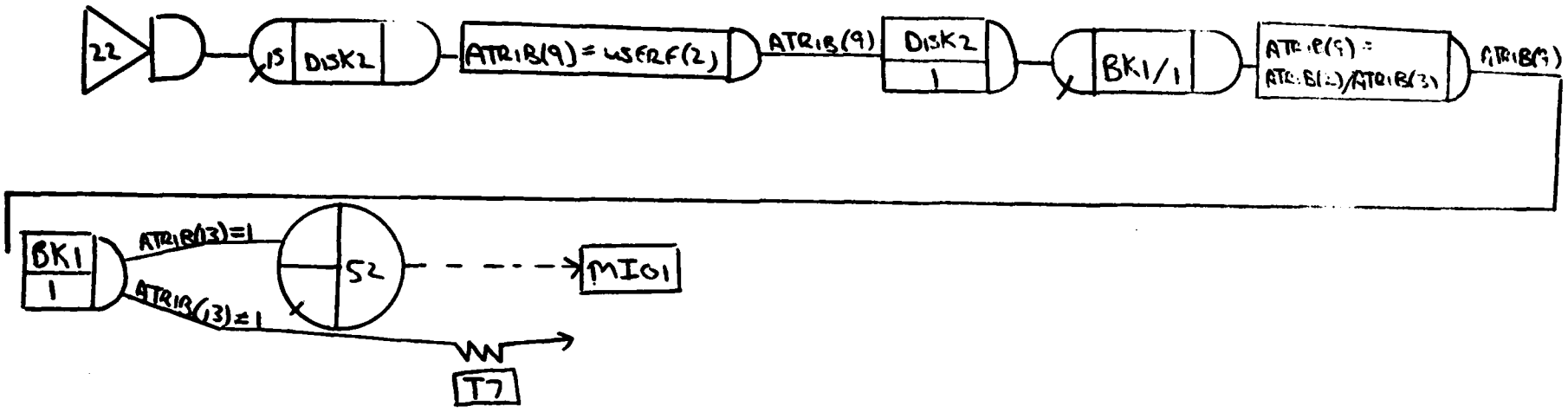

1
ATRIB(9)
ATRIB(7)=1
AS1
12
BKV1
ATRIB(11)=ATRIB(6)/2
ATRIB(11)
BK1
1
11
BBWS/1
2
T2

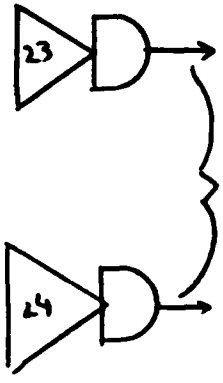

11
ATRIB(9)
BBW
1
41
MATB
Q41


2
SAME AS enter node 1 with the resource being BK2
12
SAME AS enter node 11 with the queue being Q42

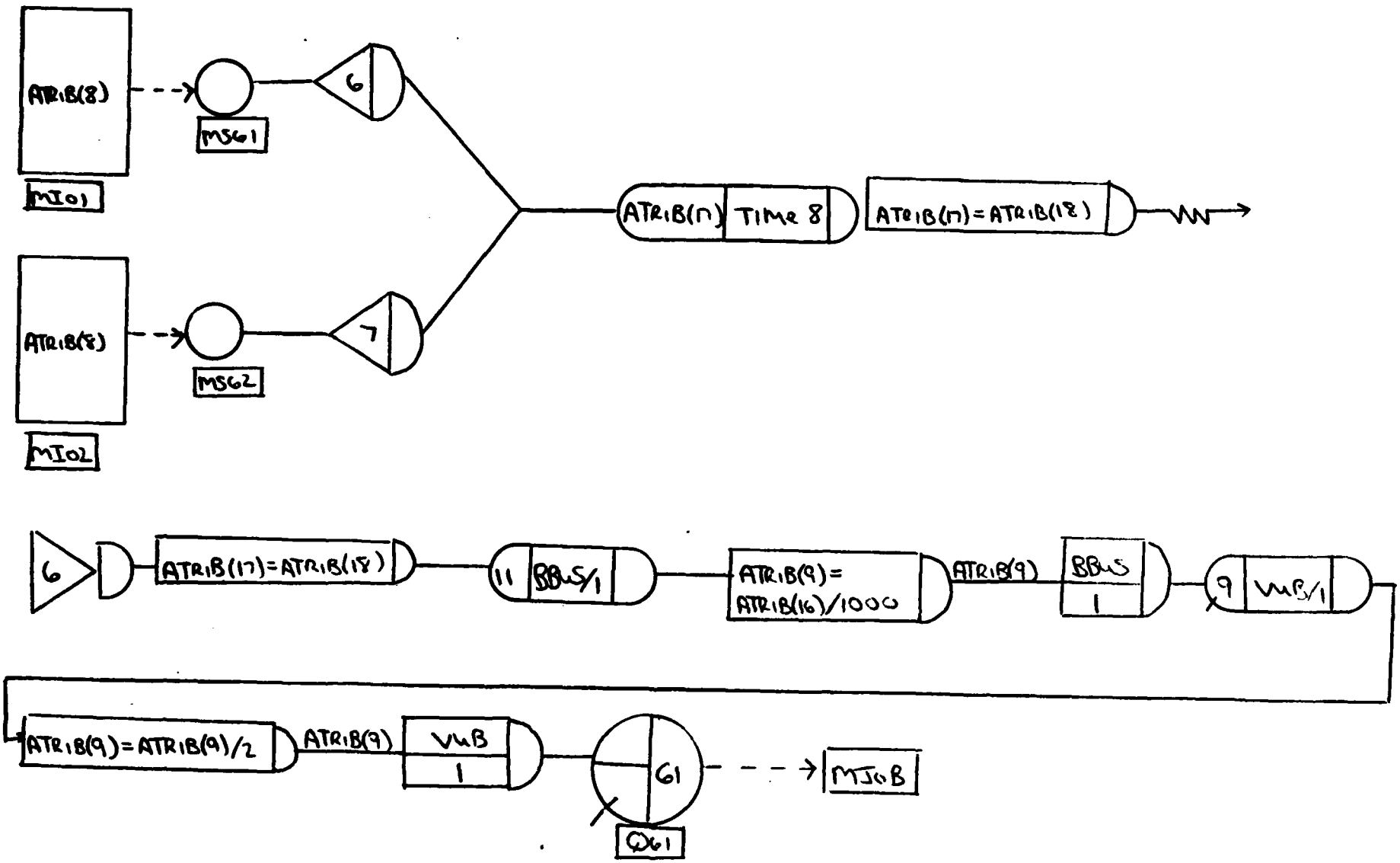

ATR:B(8)
MATB
1
BES
ATR:B(18) = TNOW - ATR:B(1)
ATR:B(17) = ATR:B(18) - ATR:B(1)
ATR:B(17)
TIME 7
ATR:B(17) = ATR:B(17)
ATR:B(10)
ATR:B(7)=1
12
BKV1
BE1
TRIAL(10,20,30)
BK1
1
ATR:B(4)=9
4
T4
ATR:B(7)=2
13
BK2/1
BE2
TRIAL(10,20,30)
BK2
1
ATR:B(4)=9
5
T5
21
DSK1
14
DCK1
ATR:B(9)=WSERF(2)
ATR:B(9)
DISK1
1
12
BK1/1
ATR:B(9) =
ATR:B(2)/ATR:B(3)
ATR:B(9)
BK1
1
ATR:B(8)=1
ATR:B(13)≠1
S1
MIO1
T6

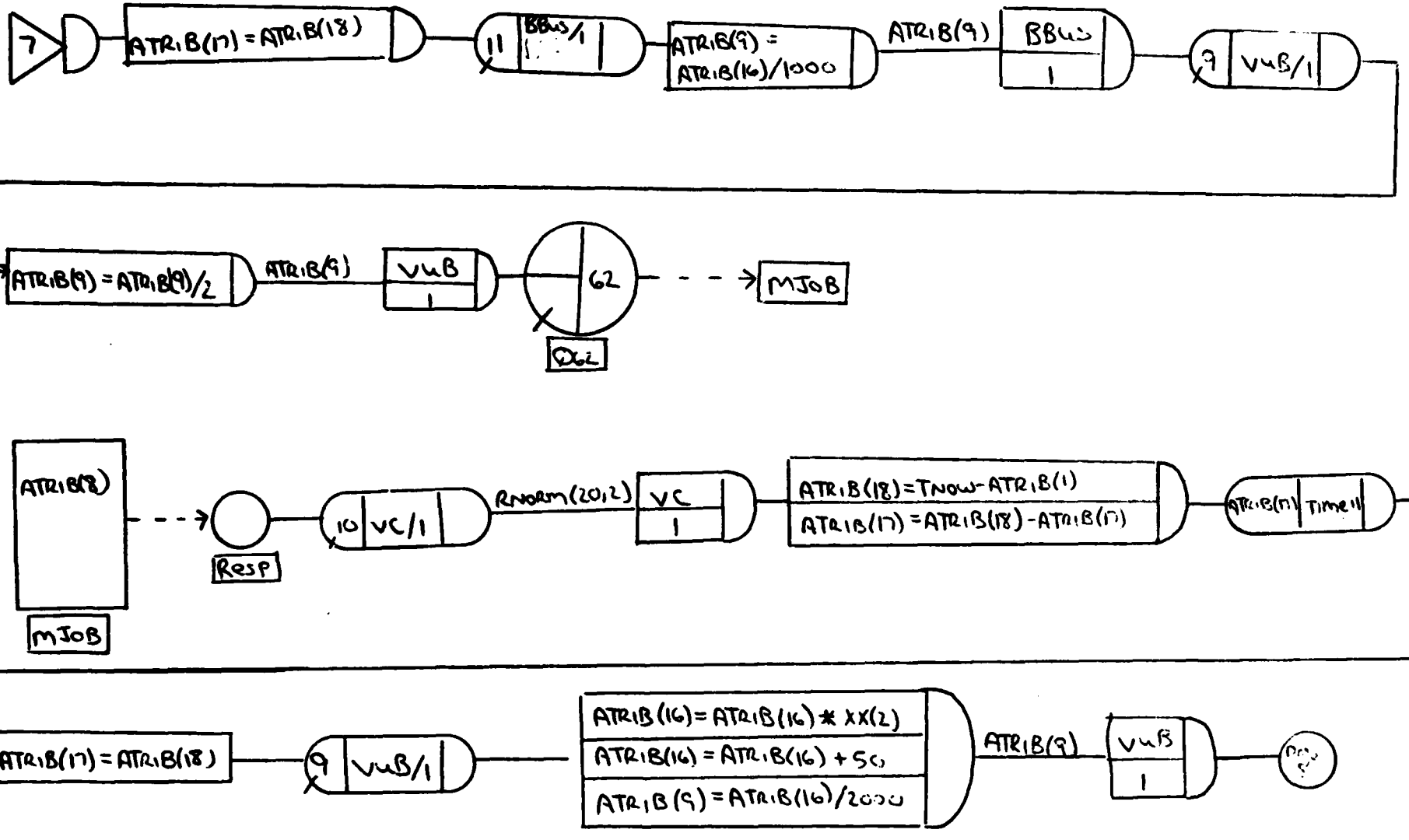
150

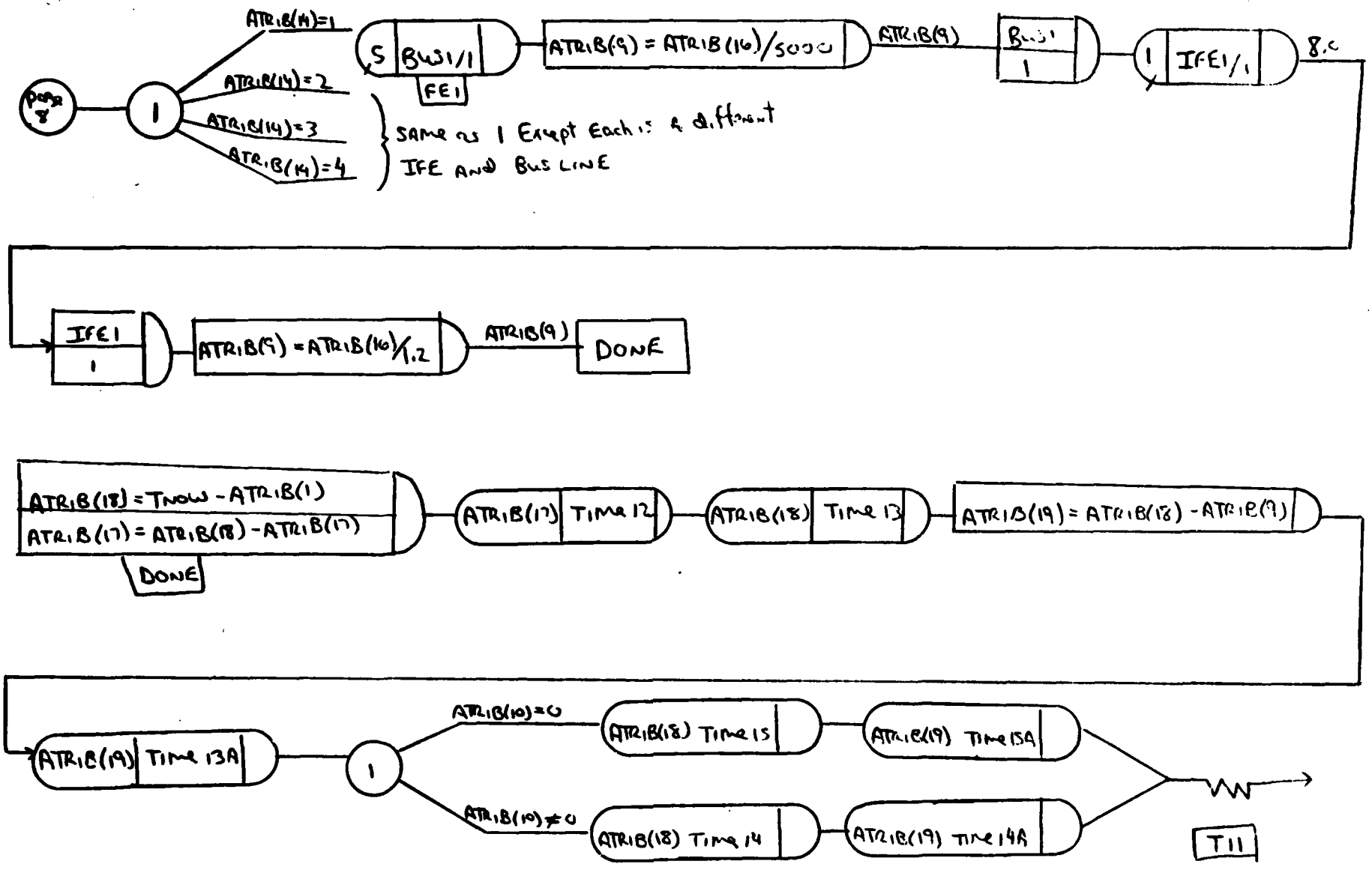




SAME as enter nodes 21 AND 22 except BK2 is used in place of BK1 AND DISKS 3 AND 4 ARE used in place of disks 1 AND 2. Queues 53 AND 54 are also used AND matched At MIO2.


ATRIB(8)
MSG1
MIO1
6
ATRIB(8)
MSG2
MIO2
7
ATRIB(11) TIME 8
ATRIB(11)=ATRIB(18)
6
ATRIB(17)=ATRIB(18)
11 BBUS/1
ATRIB(9)=
ATRIB(16)/1000
ATRIB(9)
BBUS
1
9 WB/1
ATRIB(9)=ATRIB(9)/2
ATRIB(9)
WB
1
61
MJOB
QG1


7
ATRIB(17) = ATRIB(18)
11
BBus/1
ATRIB(9) = ATRIB(16)/1000
ATRIB(9)
BBus
1
9
VuB/1
ATRIB(9) = ATRIB(9)/2
ATRIB(9)
VuB
1
62
Q62
MJOB
ATRIB(8)
mJoB
Resp
10
VC/1
RNORM(20,2)
VC
1
ATRIB(18) = TNOW - ATRIB(1)
ATRIB(17) = ATRIB(18) - ATRIB(17)
ATRIB(17)
TIME11
ATRIB(17) = ATRIB(18)
9
VuB/1
ATRIB(16) = ATRIB(16) * XX(2)
ATRIB(16) = ATRIB(16) + 50
ATRIB(9) = ATRIB(16)/2000
ATRIB(9)
VuB
1


ATRIB(14)=1
ATRIB(14)=2
ATRIB(14)=3
ATRIB(14)=4
SAME as 1 Except Each is a different IFE AND BUS LINE
S
BW31/1
FE1
ATRIB(9) = ATRIB(16)/5000
ATRIB(9)
BW31
1
IFE1/1
8.c
IFE1
1
ATRIB(9) = ATRIB(16)/1.2
ATRIB(9)
DONE
ATRIB(18) = TNOW - ATRIB(1)
ATRIB(17) = ATRIB(18) - ATRIB(17)
DONE
ATRIB(17) TIME 12
ATRIB(18) TIME 13
ATRIB(19) = ATRIB(18) - ATRIB(9)
ATRIB(19) TIME 13A
ATRIB(10)=0
ATRIB(10)≠0
ATRIB(18) TIME 15
ATRIB(19) TIME 15A
ATRIB(18) TIME 14
ATRIB(19) TIME 14A
T11

Vita

Captain Joseph L. Self was born on August 29, 1954, in Lansthul, Germany. In 1972, he graduated from the Frankfurt American High School in Frankfurt Germany. He attended the United States Air Force Academy, Colorado Springs, Colorado from which he received a Bachelor of Science degree with a major in Organizational Behavior in 1977. Upon graduation he was commissioned a Second Lieutenant in the Regular Air Force. From 1978 to 1981 he was assigned to HQ SAC Data Systems at Offutt AFB, Nebraska as a computer systems analyst and section chief. Also while in Nebraska he received a Masters degree in Business Administration from Creighton University in December 1980. He entered the Air Force Institute of Technology in June 1981.

Permanent Address:
3401 Sommerville Dr.
Montgomery, Alabama 36111

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GCS/EE/82D-32	2. GOVT ACCESSION NO. AD-A124 908	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* A BACKEND DATABASE SYSTEM FOR THE EGLIN COMPUTER NETWORK		5. TYPE OF REPORT & PERIOD COVERED MS Thesis 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Joseph L. Self Capt USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT-EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE November, 1982 13. NUMBER OF PAGES 166
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

Approved for public release: IAW AFR 190-17.

LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (AIC)
Wright-Patterson AFB OH 45433

4 JAN 8

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Backend Data Base System

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

The Directorate of Computer Sciences at Eglin AFB is in the process of upgrading its computer system to meet increased future requirements. Their upgrade includes a revision of the data base query capabilities to handle a projected larger on-line data base capability. A backend data base computer system was investigated as a possibility for meeting the anticipated increase in data base queries.

A systems analysis of the Eglin Computer Network's (ECONET) current data base query workload was accomplished to determine a baseline for future

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Ente*

workload projections. A modified Multi-backend Data Base System (MDBS - developed by David Hsiao) design was then incorporated into the ECONET and simulated with numerous structural and parametric variations. The parametric variations were to measure performance changes due to changes in workloads and the types of jobs processed. The structural variations were made to measure performance changes caused by changes in hardware configurations.

The simulation results were used to select a final backend configuration that, hopefully, would best meet the requirements of data base processing on the ECONET. A MDBS-like configuration with three backend processors was chosen.